Knowledge-Based Systems 339 (2026) 115541

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

HASNN: Hierarchical attention spiking neural network for dynamic graph representation learning

Yanglan Gan [a], Yanzu Dong [a], Wenjing Guo [a], Cairong Yan [a], Guobing Zou [b,*]

[a] School of Computer Science and Technology, Donghua University, 201620, Shanghai, China
[b] School of Computer Engineering and Science, Shanghai University, 200444, Shanghai, China

ARTICLE INFO

Keywords:
Dynamic graph neural networks
Spiking neural networks
Node state evolution
Continuous embedding preservation
Spatio-temporal attention

ABSTRACT

Dynamic graph neural networks (DGNNs) learn node representations from evolving graph structures. Recent efforts integrate spiking neural networks (SNNs) into DGNNs to capture temporal dynamics through spike-based information transmission. However, existing SNN-based DGNNs suffer from information loss caused by omitting continuous-embedding features during spike-based communication, and insufficient modeling of the temporal significance of node states. To address these limitations, we propose HASNN, a novel dynamic graph representation learning method that leverages a Hierarchical Attention Spiking Neural Network with a dual-branch spatio-temporal attention mechanism and cross-layer semantic fusion. Specifically, HASNN is a dual-path temporal framework, comprising a main channel that processes all snapshots to capture fine-grained short-term neighborhood dynamics and an auxiliary channel that subsamples snapshots with stride k to model coarse-grained long-range evolution trends. The outputs of these paths are dynamically fused using a shared attention mechanism to enhance expressive power. For each branch, HASNN adopts a dual-branch attention mechanism augmented with positional encoding to dynamically weight each node's representation at every time step. To counteract semantic discontinuity from spiking activations, HASNN retains low-order continuous embeddings post-spike to preserve representational continuity and stabilize deeper layers. Furthermore, comprehensive experiments confirm that HASNN attains state-of-the-art performance in temporal node classification across multiple dynamic graph benchmarks. The source code of the proposed HASNN is available at https://github.com/DHUDBlab/HASNN.

1. Introduction

Graphs serve as an essential data representation structure for modeling entities that interact with each other [1]. In a graph, nodes correspond to individual entities, while edges represent the interactions or relationships between them [2]. In the real world, networks are usually inherently dynamic, with their underlying structures and relationships continuously evolving over time, as illustrated in Fig. 1. Examples include social networks, where interpersonal ties form and dissolve [3], and transportation networks, characterized by fluctuating traffic and the emergence of new routes [4,5]. Effectively modeling and understanding these dynamic graphs is a challenging task. Accurately capturing these continuous transformations is crucial for temporal link prediction, identifying emerging communities, or understanding the propagation of information [6,7].

Graph neural networks (GNNs) have emerged as a prominent solution for graph representation learning from complex graph-structured data. Traditional GNNs, designed primarily for static graphs, rely on neighborhood aggregation and message passing. For example, GraphSAGE [8] performs inductive representation learning for unseen nodes through sampling, while GraphSAINT [9] enhances training efficiency by subgraph sampling. Several studies focus on pretraining [10,11] and robustness [12,13] for graph neural networks. These methods have achieved significant success in tasks such as node classification, link prediction, and graph-level prediction [14,15]. However, their static nature makes them unsuitable for dynamic graphs, which continuously evolve through the temporal addition and deletion of both nodes and edges.

To model dynamic graphs, early approaches integrate sequential models like RNNs into graph learning to capture temporal dependencies. EvolveGCN [16] adapts the parameters of graph convolutional networks over time to accommodate evolving structures. DySAT [17] introduces self-attention mechanisms to independently model both the structural and temporal dependencies of nodes. Random walk-based methods [18] aim to capture local evolutionary patterns. JODIE [19] focuses on modeling interactive dynamic graphs by learning continuous-time embedding trajectories of entities. TGAT [20] introduces an

* Corresponding author.
E-mail addresses: ylgan@dhu.edu.cn (Y. Gan), 2242840@mail.dhu.edu.cn (Y. Dong), wjguo@dhu.edu.cn (W. Guo), cryan@dhu.edu.cn (C. Yan), gbzou@shu.edu.cn (G. Zou).

https://doi.org/10.1016/j.knosys.2026.115541
Received 13 September 2025; Received in revised form 23 December 2025; Accepted 9 February 2026
Available online 18 February 2026

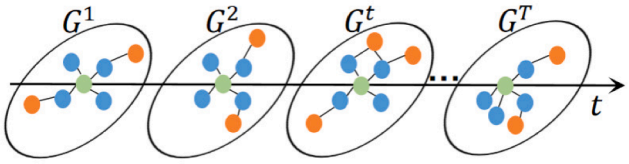

Fig. 1. A temporal graph shown as a series of snapshots, illustrating how the network structure evolves over time.

inductive framework for temporal graph embedding, which effectively integrates time encoding with structural information to model evolving graph dynamics. Despite these strengths, these methods still face challenges in capturing long-range temporal dependencies, modeling multi-scale temporal evolution [21], and preserving temporal sparsity.

Recently, Spiking Neural Networks (SNNs) have emerged as a promising paradigm for modeling complex temporal patterns, leveraging biologically-inspired event-driven computation [22,23]. Beyond energy efficiency, spiking neurons provide intrinsic temporal dynamics through membrane potential integration and leakage, acting as a continuous-time filtering mechanism that effectively captures multi-scale temporal dependencies. Specifically, the threshold-based spiking mechanism yields sparse and selective temporal coding, which aligns naturally with the irregular and non-uniform interaction patterns inherent in dynamic graphs. Unlike conventional RNNs or attention-based architectures that rely on dense, uniform-step updates, SNN-based dynamics prioritize informative event bursts while suppressing noise. This enables a more compact and robust characterization of temporal evolution, particularly when salient structural changes occur at irregular intervals.

SpikeNet [24] introduces an inductive spiking neural framework for dynamic graph learning. It adopts the Leaky Integrate-and-Fire (LIF) neuron model [25] to sample and update node states from their neighbors through discrete spike events at each time step, thereby efficiently capturing spatio-temporal dependencies in dynamic graphs while reducing both memory usage and computational overhead. While SpikeNet demonstrates strong performance, its undifferentiated spiking mechanism struggles to capture the relative importance of node information at different time points. This leads to insufficient temporal feature extraction and inadequate preservation of low-order components in node embeddings, thereby limiting its expressive capacity. SiGNN [26] leverages a Bidirectional Leaky Integrate-and-Fire (BLIF) neuron and multi-scale temporal graph sequences for enhanced dynamic graph modeling. However, SiGNN suffers from biological implausibility because of its bidirectional spiking dynamics and the excessive segmentation of temporal sequences, thereby disrupting semantic continuity and increasing model complexity. Specifically, the BLIF neuron extends the standard LIF model by introducing dual positive and negative firing thresholds together with an adaptive threshold update. This design enables SiGNN to respond to both positive and negative feature dynamics, but it also increases the number of firing events and the internal state that must be maintained, weakening the sparsity and simplicity that are typically desired in event-driven SNNs for dynamic graph scenarios. Consequently, existing SNN-based models for dynamic graphs encounter information loss caused by omitting continuous-embedding features during spike-based communication, and insufficient in modeling the temporal significance of node states [27].

To address these limitations, we introduce the Hierarchical Attention Spiking Neural Network (HASNN) for dynamic graph representation learning. HASNN integrates three complementary components to tackle the aforementioned challenges. First, a dual-path temporal framework captures multiscale temporal structures by coordinating a main channel for fine-grained global evolution and an auxiliary channel for coarse-grained local dynamics. Second, a dual-branch spatio-temporal attention mechanism, augmented with positional encoding, explicitly models the relative importance of node states across time and space. Third, a spike residual graph aggregation module fuses spike-driven high-order features with continuous low-order embeddings via cross-layer semantic fusion, mitigating semantic discontinuity and information loss inherent in spike-only communication. Together, these designs enable HASNN to learn more expressive temporal node representations and to model complex spatio-temporal dependencies in dynamic graphs.

2. Related work

2.1. Dynamic graph representation learning

The remarkable success of graph neural networks (GNNs) on static graphs has spurred growing interest in the more challenging domain of dynamic graph representation learning. Dynamic graphs, also known as temporal graphs, are characterized by nodes, edges, and their attributes evolving continuously over time. To capture this temporal evolution, most existing methods extend static GNN architectures by incorporating a temporal dimension or designing mechanisms explicitly aware of time.

A prevalent approach integrates recurrent neural networks (RNNs) as temporal modeling units. Representative approaches such as JODIE, TGN, and EvolveGCN use RNN components to update node memory states and capture the temporal continuity of node interactions. In particular, EvolveGCN evolves GCN weights via RNNs: EvolveGCN-H treats the weights as a GRU hidden state, while EvolveGCN-O generates the weights using an LSTM, thus adapting to changing graph structure. In parallel, attention-based models such as TGAT and DySAT employ self-attention mechanisms to jointly model both spatial and temporal dependencies. This enables more effective capture of intricate temporal interactions among neighboring nodes.

Despite their strong performance, these dynamic-graph GNN methods often face significant computational and memory overhead. This limits their scalability to large-scale or long-horizon dynamic graphs. Furthermore, many methods exhibit limited inductive capabilities when encountering previously unseen nodes, which significantly constrains their applicability in online or real-time dynamic graph scenarios.

2.2. SNN-based graph representation learning

Spiking neural networks (SNNs), inspired by biological neural processes, have recently become a potential solution to computational bottlenecks in dynamic graph modeling [28,29]. Unlike traditional neural networks that transmit continuous-valued signals, SNNs communicate through discrete spikes. The neurons fire only when their membrane potential exceeds a predefined threshold. This binary spiking paradigm enables energy-efficient and event-based inference, making SNNs highly suitable for neuromorphic hardware deployment.

In the graph domain, pioneering works such as SpikingGCN [30] and SpikeGCL [31] have integrated spiking mechanisms into tasks such as graph classification and contrastive learning. While SNNs inherently exhibit a strong advantage for temporal modeling, their application to dynamic graph representation learning remains underexplored. SpikeNet introduced the first SNN-based framework for dynamic graphs, leveraging the LIF neuron model to maintain spiking sparsity and energy efficiency while effectively capturing spatio-temporal dependencies. It employs an event-driven neighbor sampling strategy and incorporates an adaptive threshold mechanism, thereby enhancing both temporal modeling capability and representational flexibility. Experimental results demonstrate its superior performance across various real-world dynamic graph datasets. Building on this, SiGNN captures richer temporal evolution by sampling multi-scale structural snapshots of the graph. For feature aggregation, it adopts a BLIF neuron that is sensitive to both positive and negative temporal feature variations.

Despite the progress made by these methods, existing SNN-based approaches for dynamic graphs encounter two critical limitations. On one hand, current methods typically treat temporal information uniformly,

failing to distinguish between ordinary interactions and critical events. This uniform treatment can obscure pivotal moments that drive node state evolution in real-world scenarios, undermining the accuracy and expressiveness of the learned node representations. On the other hand, the inherently binary and discrete nature of spike signals introduces discontinuities in semantic representations. This discontinuity hampers the model's ability to capture smooth, long-range temporal dependencies crucial for holistic understanding of dynamic graph evolution.

To address these core limitations, our work proposes a novel Hierarchical Attention Spiking Neural Network (HASNN) for dynamic graph representation learning. We enable fine-grained temporal importance modeling through temporally aware attention and global-local fusion. Furthermore, we mitigate semantic discontinuity using a gated and residual design within our spike residual aggregation, thereby enhancing both the expressiveness and temporal continuity of dynamic graph representations.

3. Preliminary

3.1. Problem definition

Given a sequence of dynamic graph snapshots $G = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$ over T time steps, each snapshot at time t is denoted as $G^{(t)} = (V, E^{(t)}, X^{(t)})$. The dynamic graph shares a fixed node set $V = \{v_1, \ldots, v_n\}$ across all time steps, while the edge set $E^{(t)} \subseteq V \times V$ and node feature matrix $X^{(t)} \in \mathbb{R}^{n \times d}$ may vary over time. Let $N_v^{(t)}$ denote the neighborhood of node $v \in V$ at time t. Dynamic graph representation learning aims to learn a mapping $f : \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\} \to \{H^{(1)}, H^{(2)}, \ldots, H^{(T)}\}$, where each $H^{(t)} \in \mathbb{R}^{n \times d}$ is the node embedding matrix at time step t, representing the state of the entire graph in terms of node embeddings. These embeddings can then be directly utilized in downstream tasks such as dynamic node classification and temporal link prediction.

3.2. Leaky integrate-and-fire (LIF) model

The LIF model is a classical spiking neuron model used to simulate the fundamental electrical behavior of biological neurons. Its core mechanism involves the integration and leakage of membrane potential $V(t)$, governed by the following differential equation:

$$\tau_m \frac{dV(t)}{dt} = -\big(V(t) - V_{\text{reset}}\big) + I(t) \tag{1}$$

where τ_m is the membrane time constant controlling the leak rate, V_{reset} is the reset potential, and $I(t)$ denotes the input current to the neuron at time t.

To discretize this process for computational implementation, the equation [32] can be represented as:

$$V^t = V^{t-1} + \frac{1}{\tau_m}\big(-(V^{t-1} - V_{\text{reset}}) + I^t\big) \tag{2}$$

When the membrane potential V^t exceeds the firing threshold V_{th}, the neuron emits a spike and resets its potential:

$$S^t = \begin{cases} 1 & V^t \geq V_{\text{th}} \\ 0 & V^t < V_{\text{th}} \end{cases} \tag{3}$$

where $S^t \in \{0, 1\}$ indicates whether a spike is emitted at time t. If no spike occurs ($S^t = 0$), the membrane potential remains. Conversely, upon spike emission ($S^t = 1$), the membrane potential is immediately reset to V_{reset} :

$$V^t = \begin{cases} V^t & S^t = 0 \\ V_{\text{reset}} & S^t = 1 \end{cases} \tag{4}$$

This reset mechanism ensures that spikes are only generated when the neuron receives sufficient stimulation, thereby mimicking the all-or-nothing firing behavior characteristic of biological neurons.

4. Proposed method

4.1. Overview of HASNN

To effectively capture spatio-temporal node evolution in dynamic graphs, we propose the Hierarchical Attention Spiking Neural Network (HASNN) framework. As illustrated in Fig. 2, HASNN consists of three key modules: hierarchical neighborhood sampling, spike residual graph aggregation, temporal attention channel aggregation.

Firstly, we introduce a hierarchical neighborhood sampling scheme with a fine-grained main channel and a coarse-grained auxiliary channel. The main channel samples every snapshot to preserve short-term continuity, ensuring that local structural changes are captured with high resolution. The auxiliary channel subsamples the sequence with stride k, reducing redundant computation while summarizing long-range dependencies. This dual-branch design resolves the trade-off between high-resolution sensitivity and computational efficiency, providing HASNN with both detailed and global views of the evolving graph.

Subsequently, for each sampled neighborhood, we perform spike residual graph aggregation, employing a LIF spiking aggregator to encode abrupt temporal changes via sparse spikes. However, purely spike-based communication can lead to semantic discontinuity and vanishing gradients in deep layers. To counter this, we fuse the continuous low-order embeddings from previous layers with the high-order spike-driven features through cross-layer residual connections. This fusion preserves semantic stability and enhances representational continuity. Crucially, it facilitates the training of deep SNN architectures, offering a novel spike-aware residual mechanism that distinguishes our model from existing approaches.

Finally, we introduce a dual-branch temporal attention mechanism to adaptively weight node embeddings across time. The position-aware branch adds sinusoidal positional encodings before a linear and softmax pipeline, thereby emphasizing critical time steps. In contrast, the position-free branch applies the same projection directly to raw embeddings, focusing on feature dynamics that are independent of absolute time indices. By computing softmax-normalized attention weights in both paths and fusing them with learnable coefficients, HASNN effectively addresses the challenge of uneven temporal importance, enabling semantic alignment and cross-snapshot fusion across multiple temporal scales, rather than treating all time steps uniformly as in prior SNN-based dynamic graph methods.

4.2. Hierarchical neighborhood sampling

As shown in Fig. 3, to capture structural evolution of a dynamic graph at multiple temporal scales, we introduce a hierarchical two-level neighborhood sampling scheme. For each target node v in the dynamic graph $G^m = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$, where $G^{(t)}$ denotes the graph snapshot at time step t, we construct two synchronized temporal input paths.

The main channel begins at the target node v and, at each time step $t \in \{1, \ldots, T\}$, samples its r-hop neighborhood from $G^{(t)}$, thereby constructing the local subgraph $N_v^{(t)}$. We thus obtain the neighborhood-subgraph sequence $\{N_v^{(t)}\}_{t=1}^{T}$, which enables continuous monitoring of immediate neighbor dynamics for sensitive detection of short-term structural changes.

In parallel, the auxiliary channel operates on a temporally downsampled version of the original sequence to capture long-range structural transitions with reduced computational cost. We introduce a fixed sampling stride k (set $k = 2$ in our experiments) and define the index set $S = \{1, 1+k, 1+2k, \ldots, 1+(T'-1)k\}$, where $T' = \left\lfloor \frac{T-1}{k} \right\rfloor + 1$ denotes the length of the downsampled sequence. The auxiliary channel first samples the dynamic graph sequence with step k, to reduce computational cost while still preserving long-range temporal transitions across the graph evolution, and constructs the subsequence $G^a = \{G^{(t)} \mid$

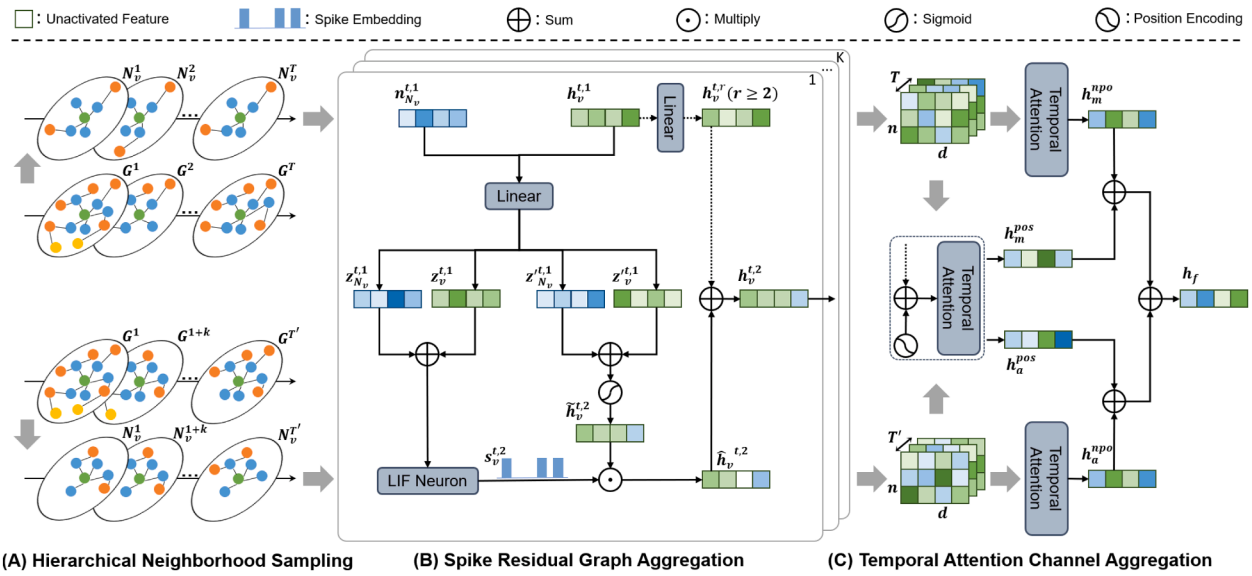

Fig. 2. The framework of HASNN. HASNN consists of three major modules, including hierarchical neighborhood sampling, spike residual graph aggregation and temporal attention channel aggregation modules.

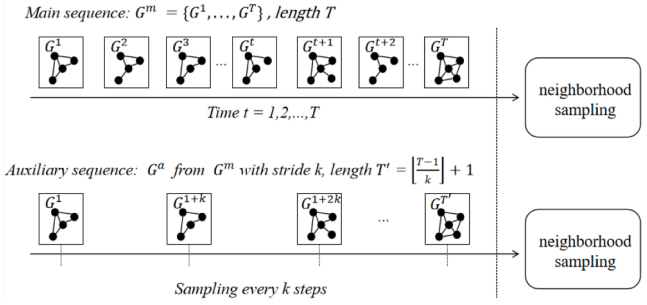

Fig. 3. Illustration of hierarchical two-level temporal sampling. The main path processes all snapshots for fine-grained dynamics, while the auxiliary path samples every $k-th$ snapshot to capture coarse-grained patterns.

$t \in S\}$. Based on G^a, it then extracts the corresponding neighbor sets for node v: $N'_v = \{N_v^{(t)} \mid t \in S\}$.

In other words, G^a is obtained from G^m by sampling every k-th snapshot, and T' is exactly the number of sampled time steps. Different values of k control the temporal granularity of the auxiliary channel. A smaller value of k retains finer short-term dynamics, while a larger value focuses on coarser long-range trends. In our experiments, we set $k = 2$ to balance effectiveness and efficiency.

The outputs of both channels are processed in parallel by subsequent graph aggregation layers. The main channel provides high-resolution, short-term neighborhood dynamics, while the auxiliary channel captures coarse-grained, long-term structural dependencies. This hierarchical integration enables simultaneous modeling of rapid fine-grained changes and slower coarse-grained transitions, generating rich spatio-temporal features for the downstream spiking aggregation and temporal attention modules.

4.3. Spike residual graph aggregation

For the resulting neighborhood of each channel, we subsequently conduct spike residual graph aggregation (SRGA). First, we employ an LIF-based spiking aggregator to encode abrupt temporal changes via sparse spikes. Purely spike-based communication, however, can cause semantic discontinuity between layers and vanishing gradients in deep architectures. To mitigate these issues, we introduce cross-layer residual fusion, which reuses continuous low-order embeddings from previous layers while keeping the spike signal as a binary gate on the newly computed activations. In this way, spike-driven features capture event-triggered high-order updates, whereas the residual path provides a continuous semantic baseline, enhancing spatio-temporal embeddings and stabilizing training in our deep SNN architecture.

From the second layer, each layer performs the following steps:

Neighborhood Aggregation. For a node v at time t, the neighborhood N_v^t is aggregated through mean/max/sum operations:

$$n_{N_v}^{t,(k)} = AGG(h_u^{t,(k)}, \forall u \in N_v^t) \in \mathbb{R}^{n \times d_k} \tag{5}$$

where $n_{N_v}^{t,(k)}$ denotes the aggregated feature representation of the sampled neighbors of node v at layer k and time step t, and $AGG(\cdot)$ represents aggregation function.

Dual Mapping. We separately project the aggregated neighborhood features $n_{N_v}^{t,(k)}$ and the self-embedding $h_v^{t,(k)}$ through linear mappings:

$$z_{N_v}^{t,(k)} = n_{N_v}^{t,(k)} W_h^k, \quad z_v^{t,(k)} = h_v^{t,(k)} B_h^k \tag{6}$$

$$z'^{t,(k)}_{N_v} = n_{N_v}^{t,(k)} W'^k_h, \quad z'^{t,(k)}_v = h_v^{t,(k)} B'^k_h \tag{7}$$

Here, z and z' are intermediate embeddings fed into the LIF spiking neuron and the nonlinear activation path, respectively. W, B and W', B' are layer-specific trainable parameters. We then feed these intermediate embeddings into the LIF spiking neuron and the subsequent nonlinear activation, respectively.

$$s_v^{t,(k+1)} = LIF(z_{N_v}^{t,(k)} + z_v^{t,(k)}) \tag{8}$$

$$\tilde{h}_v^{t,(k+1)} = \sigma(z'^{t,(k)}_{N_v} + z'^{t,(k)}_v) \tag{9}$$

where LIF denotes the binary spike generation process that outputs the spike vector $s_v^{t,(k+1)}$, and σ is a nonlinear activation function producing the continuous node embedding $\tilde{h}_v^{t,(k+1)}$.

Cross-layer Residual Fusion. To address the semantic gap between discrete spiking activations and continuous feature representations, we introduce residual fusion by reusing low-level embeddings $h_v^{t,r}$ from previous layers. This promotes temporal consistency, stabilizes feature representations, and facilitates gradient propagation for long-term dynamic modeling. This fusion mechanism is formalized as:

$$h_v^{t,r} = h_v^{t,(k)} W_r^{(k)}, W_r^{(k)} \in \mathbb{R}^{d_k \times d_{k+1}} \tag{10}$$

To ensure continuous propagation of low-level semantics while retaining event-driven updates, we integrate spiking signals with candidate

activations through element-wise gating and augment them with residual mapping:

$$\hat{h}_v^{t,(k+1)} = s_v^{t,(k+1)} \odot \tilde{h}_v^{t,(k+1)} + h_v^{t,r} \tag{11}$$

Finally, the output of the k^{th} aggregator layer is obtained by applying dropout to the fused representation:

$$h^{t,(k+1)} = Dropout(\hat{h}^{t,(k+1)}) \tag{12}$$

For the initial aggregation layer ($k = 1$), residual fusion is omitted due to the absence of preceding spiking activations. The output simplifies to gated modulation only:

$$\hat{h}^{t,(k+1)} = s_v^{t,(k+1)} \odot \tilde{h}_v^{t,(k+1)} \tag{13}$$

Thus, the module SRGA improves aggregation efficiency through sparse spike activation, while simultaneously leveraging cross-layer residual fusion to explicitly retain low-level semantic information. This approach not only preserves temporal continuity in node embeddings but also reinforces the integration of fine-grained local details with global structural evolution patterns. After spatial aggregation across all nodes in both main and auxiliary channels, the spatial modeling phase is completed. The framework subsequently transitions to temporal modeling to capture dynamic evolution over time. Then dual-path spatio-temporal embedding tensors are generated: the main channel $H^m \in \mathbb{R}^{T\times n\times d}$ and the auxiliary channel $H^a \in \mathbb{R}^{T'\times n\times d}$.
where T denotes the total number of time steps in the main sequence, $T' = \left\lfloor \frac{T-1}{k} \right\rfloor + 1$ is the number of snapshots sampled in the auxiliary sequence with stride k, n denotes the number of nodes, and d the feature dimensionality. This dual-path embedding formulation ensures temporal structural completeness across both fine- grained and coarse-grained sequences.

4.4. Temporal attention channel aggregation

To model the temporal importance of node representations, we integrate an attention mechanism into the SNN backbone. Specifically, to better capture heterogeneous temporal patterns in dynamic graphs, we introduce a dual-branch temporal attention module with two distinct pathways: a position-encoded branch and a non-positional branch. Dynamic node interactions often exhibit both order-sensitive effects (e.g., recency or stage-dependent behaviors) and patterns that are largely insensitive to absolute time indices (e.g., similar interaction motifs reappearing at different time steps). The position-encoded branch emphasizes absolute temporal order and global trends, whereas the non-positional branch focuses on content-based similarity across time without relying on specific time indices. By fusing the outputs of these two branches with learnable coefficients w_{pos} and w_{npo} (with $w_{pos} + w_{npo} = 1$), HASNN adaptively balances order-aware and order-agnostic temporal information in both the main and auxiliary channels, thereby enhancing the expressiveness of the learned dynamic graph representations.

In the position-encoded branch, we first apply positional encoding (PE) to the input sequence. Subsequently, a linear transformation layer processes this encoded input, followed by softmax normalization to compute the temporal attention weights. These weights dynamically assign varying importance to node embeddings across different time steps, enabling adaptive temporal representation learning. This mechanism not only enhances the model's capacity to capture temporal heterogeneity within node representations but also facilitates semantic alignment and cross-snapshot information fusion. The temporal attention module then processes the 3D embedding tensors H^m and H^a, respectively from the main and auxiliary channels, utilizing either position-aware encoding or position-free encoding.

The temporal positional encoding is defined using sinusoidal functions:

$$PE(t, 2i) = sin(\frac{t}{10000^{\frac{2i}{d}}}) \tag{14}$$

$$PE(t, 2i+1) = cos(\frac{t}{10000^{\frac{2i}{d}}}) \tag{15}$$

where t denotes the time step, i denotes the dimension index, and d represents the total dimensionality of the positional encoding vector.

We denote the original time-specific node embeddings as $H_{c,t} \in \mathbb{R}^{n\times d} (c \in \{m, a\})$. After preparing the input embeddings with or without positional encoding, the attention module handles the position-encoded path and non-position-encoded path as follows:

$$\tilde{H}_{c,t}^{pos} = H_{c,t} + PE_t \quad \tilde{H}_{c,t}^{npo} = H_{c,t} \tag{16}$$

Temporal attention scores are then computed through branch-specific linear projections. Specifically, for the position-encoded path, a learnable projection $W_a \in \mathbb{R}^{d\times 1}$ is applied to compute the attention coefficient for each time step. For the non-position-encoded path, another projection is used.

$$e_{c,t}^{pos} = \tilde{H}_{c,t}^{pos} W_a \quad e_{c,t}^{npo} = \tilde{H}_{c,t}^{npo} W_b \tag{17}$$

Next, the attention logits $\{e_{c,t}\}_{t=1}^{T_c}$ (where $T_m = T$ and $T_a = T'$) are normalized along the temporal dimension using the softmax function:

$$A_{c,t}^{pos} = \frac{\exp(e_{c,t}^{pos})}{\sum_{j=1}^{T_c} \exp(e_{c,j}^{pos})}, \quad A_{c,t}^{npo} = \frac{\exp(e_{c,t}^{npo})}{\sum_{j=1}^{T_c} \exp(e_{c,j}^{npo})}, \tag{18}$$

where $A_{c,t}^{pos}, A_{c,t}^{npo} \in \mathbb{R}^{n\times 1}$ denote node-wise temporal attention weights at time t.

These normalized attention weights $A_{c,t}^{pos}$ and $A_{c,t}^{npo}$ are used to perform a weighted sum over the temporal features, generating the final time-aggregated representations:

$$h_c^{pos} = \sum_{t=1}^{T_c} A_{c,t}^{pos} \odot \tilde{H}_{c,t}^{pos}, h_c^{npo} = \sum_{t=1}^{T_c} A_{c,t}^{npo} \odot \tilde{H}_{c,t}^{npo} \tag{19}$$

where $\odot$ denotes Hadamard product. These representations of the positional and non-positional encoding paths are fused with learnable weights w_{pos} and w_{npo}, satisfying $w_{pos} + w_{npo} = 1$:

$$h_c = w_{pos} h_c^{pos} + w_{npo} h_c^{npo} \tag{20}$$

The output h_c is computed independently for the main and auxiliary channels($c \in \{m, a\}$. These channel-specific embeddings are aggregated to produce the final node representation:

$$h_f = Mean(h_m, h_a) \tag{21}$$

Finally, the output h_f is fed into a downstream classifier. The module simultaneously preserves two distinct temporal learning pathways, high-level channel interactions, and time-step specific attention mechanisms. By selectively emphasizing critical temporal segments, HASNN exhibits the spatio-temporal representation capabilities for diverse dynamic graph learning tasks.

5. Experiment

In this section, we conduct comprehensive experiments on three real-world dynamic graph datasets to evaluate the effectiveness of the proposed method on the node classification task. First, we introduce the datasets and evaluation metrics. Subsequently, we compare our model against a range of state-of-the-art baseline methods. Finally, we perform ablation studies and parameter sensitivity analyses to validate the effectiveness and robustness of each module within our framework.

5.1. Datasets

We benchmark HASNN on temporal node classification across three real-world dynamic-graph datasets-DBLP, Tmall, and Patent.

DBLP: This dataset models co-authorship as a dynamic graph, where nodes represent authors and edges signify co-authorship in the same paper. It is divided into 27 temporal snapshots, comprising 28,085 nodes

Table 1
Dataset statistics.

Attributes	DBLP	Tmall	Patent
Nodes	28,085	577,314	2,738,012
Edges	236,894	4,807,545	13,960,811
Classes	10	5	6
Time steps	27	186	25

and 236,894 edges. Each node is labeled with one of 10 research areas, corresponding to the author's field.

Tmall: This dataset represents a dynamic user-item interaction graph from an e-commerce platform. Nodes represent users and products, while edges denote either a purchase or Browse behavior. The graph contains 186 temporal snapshots, with 577,314 nodes and 4,807,545 interaction edges, covering 5 distinct product categories.

Patent: This is a dynamic patent citation network, where each node represents a patent document and edges indicate citation relationships. The dataset comprises 25 yearly snapshots, encompassing 2,738,012 nodes and 13,960,811 citation edges. Each node is classified into one of six major technology domains.

These three datasets exhibit significant differences in terms of node scale, edge density, number of categories, and temporal span. This diversity enables us to thoroughly assess the scalability, robustness, and general applicability of the proposed model across dynamic graphs evolving over both short and long terms, and ranging from small to large scales (Table 1).

5.2. Baselines

Seven existing models are selected as the baselines, which includes static graph embeddings methods (Deepwalk [33] and Node2Vec [34]), Classical Dynamic Graph methods (JODIE [19], EvolveGCN [16] and TGAT [20]), and SNN-based dynamic methods (SpikeNet [24], SiGNN [26]). All baselines are introduced as follows:

- DeepWalk: samples short random-walk sequences, then applies Skip-Gram to learn embeddings that encode neighborhood structure.
- Node2Vec: extends DeepWalk with biased random walks to balance local neighborhood preservation and global network exploration.
- JODIE: uses coupled recurrent neural networks to produce continuous-time embedding trajectories for user-item interactions.
- EvolveGCN: dynamically updates GCN parameters through recurrent units (e.g., GRU / LSTM), adapting structural filters to topology evolution.
- TGAT: learns joint structural-temporal representations through self-attention mechanisms with time-encoding kernels for inductive settings.
- SpikeNet: integrates LIF neurons into graph convolution layers for spike-based temporal dynamics modeling.
- SiGNN: extends SpikeNet with bidirectional spiking and multi-scale snapshot sampling to enrich temporal modeling.

5.3. Performance comparison

To validate the performance of the Hierarchical Attention Spiking Neural Network (HASNN), we conduct a comparative analysis against the above seven representative methods. All methods are evaluated on three widely-used benchmarks (DBLP, TMALL, Patent) for temporal node classification task. The performance is measured by both micro-F1 and macro-F1 scores, where micro-F1 quantifies overall classification accuracy by aggregating across all nodes, thus emphasizing performance on majority classes, and macro-F1 averages per-class F1 scores to reveal effectiveness in handling rare or minority classes. To assess robustness and data efficiency, we vary the training ratio (40%, 60%, 80% of labeled nodes). Lower ratios test generalization ability with limited labels, while higher ratios demonstrate scalability with increased supervision.

Table 2 shows the performance comparison results of these methods. HASNN achieves state-of-the-art performance on DBLP, Tmall, and Patent, attaining the highest Macro-F1 and Micro-F1 and consistently surpassing all baselines-including SpikeNet and SiGNN-across varying training ratios. These results demonstrate robust dynamic node representation learning and the scalability of the spiking-attention framework, further confirming its state-of-the-art performance. Key findings are summarized as follows:

- Robustness and scalability. As the training ratio increases from 40% to 80%, all methods show performance gains. Notably, HASNN exhibits the most substantial improvements. This indicates its strong ability to leverage available supervision and generalize across data scales, making it well-suited for both low-resource and fully supervised scenarios.
- Importance of temporal modeling. The three dynamic models (JODIE, EvolveGCN, TGAT) consistently outperform static baselines (DeepWalk, Node2Vec) under all settings, emphasizing the necessity of modeling temporal evolution in dynamic graphs. Among them, HASNN leads by a clear margin, further confirming its advantage in modeling complex temporal structures.
- Advantage of SNN-based design. Compared to other spiking neural network approaches, HASNN consistently outperforms SpikeNet and SiGNN, demonstrating that integrating spiking dynamics with attention mechanisms and temporal abstraction offers a powerful paradigm for dynamic graph learning.

5.4. Ablation study

To validate the efficacy of various components in our model, we conduct extensive ablation studies. On three datasets (DBLP, TMall and Patent) under three training splits (40%, 60%, and 80%) , we compare HASNN with its six variants:

- w/o SRGA&TA: remove both spike-residual fusion and temporal attention
- w/o SRGA: remove both the cross-layer residual fusion and the spike-gated feature activation.
- w/o Residual: remove only the cross-layer residual fusion while retaining the spike-gated feature activation.
- w/o TA: remove both attention branches entirely
- w/o TA-npo: remove only the non-position-encoded branch attention branch
- w/o TA-pos: remove only the position-encoded attention branch

Table 3 presents the performance outcomes of the HASNN model and its six variants. From the table, we observe that the HASNN model outperforms all ablation variants, affirming the essential role of each component. Compared to HASNN, the variant without temporal attention and cross-layer residual fusion exhibited the poorest performance, with Macro-F1 and Micro-F1 scores decreasing by approximately 3.5∼4.5% and 3.3∼4.0%, respectively. The result indicate that temporal attention and cross-layer residual fusion contribute significantly to the model's performance. Removing SRGA alone reduces performance by about 3%, while removing TA lead to performance decrease by about 2% , demonstrating that the two modules significantly enhance performance. Removing only the residual connection decrease the performance by 1.2∼1.9%. Similarly, removing all attention resulted in a performance decrease about 0.9∼1.5% in Macro-F1 and 1.0∼1.9% in Micro-F1. Conversely, removing only the position-encoded branch or the non-position-encoded branch (w/o TA-pos) respectively decrease 0.3∼2% and 0.8∼1.4% in Macro-F1, underscoring the complementary information derived from explicit temporal ordering and broader contextual information.

For the proposed HASNN, the fusion of fine- and coarse-grained temporal embeddings is important for capturing both short-term dynamics

Table 2
Performance comparison of HASNN with baseline models on the node classification (All values are reported in percentage: %, Tr. ratio: training ratio).

Dataset	Metrics	Tr. ratio	DeepWalk	Node2Vec	JODIE	EvolveGCN	TGAT	SpikeNet	SiGNN	HASNN
DBLP	**Macro-F1**	**40%**	67.08	66.07	66.73±1.0	67.22±0.3	71.18±0.4	70.71±0.38	74.66±0.26	**76.53±0.42**
		60%	67.17	66.81	67.32±1.1	69.78±0.8	71.74±0.5	72.13±0.22	76.68±0.36	**79.21±0.35**
		80%	67.12	66.93	67.53±1.3	71.20±0.7	72.15±0.3	73.91±0.49	78.74±0.22	**82.13±0.32**
	Micro-F1	**40%**	66.53	66.80	68.44±0.6	69.12±0.8	71.10±0.2	72.00±0.27	75.36±0.18	**77.25±0.47**
		60%	66.89	67.37	68.51±0.8	70.43±0.6	71.85±0.4	73.21±0.31	77.02±0.20	**79.57±0.23**
		80%	66.38	67.31	68.80±0.9	71.32±0.5	73.12±0.3	74.53±0.50	79.16±0.21	**82.45±0.35**
Tmall	**Macro-F1**	**40%**	49.09	54.37	52.62±0.8	53.02±0.7	56.90±0.6	59.38±0.12	62.47±0.20	**63.29±0.16**
		60%	49.29	54.55	54.02±0.6	54.99±0.7	57.61±0.7	61.14±0.12	64.40±0.31	**65.12±0.21**
		80%	49.53	54.58	54.17±0.2	55.78±0.6	58.01±0.7	62.89±0.34	66.14±0.28	**67.23±0.31**
	Micro-F1	**40%**	57.11	60.41	58.36±0.5	59.96±0.7	62.05±0.5	63.50±0.13	66.43±0.07	**67.27±0.10**
		60%	57.34	60.56	60.28±0.3	61.19±0.6	62.92±0.4	65.03±0.14	68.15±0.14	**69.12±0.14**
		80%	57.88	60.66	60.49±0.3	61.77±0.6	63.32±0.7	66.53±0.33	69.63±0.08	**70.24±0.26**
Patent	**Macro-F1**	**40%**	72.32±0.9	69.01±0.9	77.57±0.8	79.67±0.4	81.51±0.4	83.53±0.6	84.45±0.04	**85.10±0.30**
		60%	72.25±1.2	69.08±0.9	77.69±0.6	79.76±0.5	81.56±0.6	83.85±0.7	84.81±0.05	**85.24±0.38**
		80%	72.05±1.1	68.99±1.0	77.67±0.4	80.13±0.4	81.57±0.5	83.90±0.6	84.91±0.03	**85.38±0.32**
	Micro-F1	**40%**	71.57±1.3	68.14±0.9	77.64±0.7	79.39±0.5	80.79±0.7	83.48±0.8	84.41±0.02	**85.11±0.31**
		60%	71.53±1.0	68.20±0.7	77.89±0.5	79.95±0.3	80.81±0.6	83.80±0.7	84.78±0.04	**85.27±0.37**
		80%	71.38±1.2	68.10±0.5	77.93±0.4	80.01±0.3	80.93±0.6	83.88±0.9	84.89±0.03	**85.40±0.27**

Table 3
HASNN module ablation study.

Dataset	Metrics	Tr. ratio	w/o SRGA&TA	w/o SRGA	w/o Residual	w/o TA	w/o TA-pos	w/o TA-npo	HASNN
DBLP	**Macro-F1**	**40%**	73.10	74.31	75.60	75.27	76.07	76.23	**76.53**
		60%	74.50	75.89	77.57	77.28	77.95	78.31	**79.21**
		80%	77.03	78.76	80.62	79.55	80.13	80.63	**82.13**
	Micro-F1	**40%**	73.03	74.65	76.20	75.91	76.45	76.67	**77.25**
		60%	75.10	76.25	78.13	77.88	78.21	78.67	**79.57**
		80%	73.32	79.41	81.01	80.54	80.89	81.51	**82.45**
Tmall	**Macro-F1**	**40%**	59.64	60.33	62.44	62.67	63.01	63.10	**63.29**
		60%	61.66	62.18	64.38	64.14	64.37	64.67	**65.12**
		80%	63.86	64.43	66.23	66.32	66.53	66.70	**67.23**
	Micro-F1	**40%**	64.79	65.57	66.52	66.47	66.54	66.75	**67.27**
		60%	65.83	66.32	68.19	67.98	68.21	68.50	**69.12**
		80%	67.78	68.31	69.56	69.80	69.87	70.12	**70.24**
Patent	**Macro-F1**	**40%**	83.82	84.02	84.63	84.45	84.76	84.79	**85.10**
		60%	84.03	84.21	84.78	84.58	85.01	85.11	**85.24**
		80%	84.21	84.45	84.92	84.62	85.12	85.25	**85.38**
	Micro-F1	**40%**	83.84	84.04	84.61	84.57	84.77	84.78	**85.11**
		60%	84.01	84.22	84.80	84.59	85.02	85.13	**85.27**
		80%	84.20	84.46	84.93	84.64	85.13	85.24	**85.40**

and long-range dependencies in dynamic graphs. We compare four fusion strategies: max, mean, concat, and attention. As shown in Table 4, the mean fusion method consistently outperformed the other strategies across multiple datasets. This suggests that mean fusion provides a stable and unbiased integration of temporal information, effectively preserving complementary patterns from both high-frequency and low-frequency evolutions.

Table 4
The ablation study of different main-auxiliary fusion strategies.

Dataset	Metrics	Tr. ratio	Mean	Max	Concat	Attention
DBLP	**Macro-F1**	**40%**	**76.53**	75.51	76.26	75.59
		60%	**79.21**	77.97	78.30	78.42
		80%	**82.13**	80.51	81.02	80.79
	Micro-F1	**40%**	**77.25**	75.95	76.44	76.63
		60%	**79.57**	77.93	78.68	78.70
		80%	**82.45**	80.81	81.24	81.15
Tmall	**Macro-F1**	**40%**	**63.29**	62.13	63.11	62.77
		60%	**65.12**	63.96	64.44	64.11
		80%	**67.23**	65.87	66.83	66.34
	Micro-F1	**40%**	**67.27**	65.65	66.89	66.80
		60%	**69.12**	67.89	68.99	68.67
		80%	**70.24**	69.23	70.03	69.85

5.5. Hyperparameter analysis

5.5.1. Effect of temporal stride k in the auxiliary channel

HASNN incorporates a dual-path temporal architecture wherein the main channel processes every graph snapshot at native resolution, while the auxiliary channel samples snapshots at coarser intervals determined by a configurable stride parameter k. This synergistic design allows the model to construct multi-scale temporal receptive fields, enabling it to capture both fine-grained short-term variations and coarse-grained long-term trends in evolving graph structures (Fig. 4).

To determine the optimal temporal granularity, we conduct a performance comparison with the following stride configurations $k \in \{0, 1, 2, 3, 4\}$ in HASNN's auxiliary channel. Specifically, when $k = 0$, the auxiliary channel is disabled, which serves as a baseline for single-scale temporal modeling. When $k = 1$, both channels are used, but the auxil-

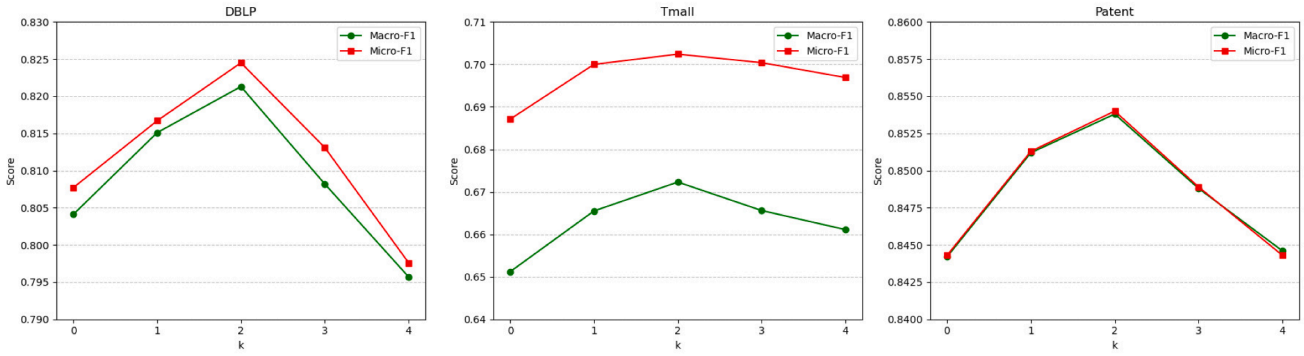

Fig. 4. The choice of auxiliary channel stride.

iary path operates at the same resolution as the main path. When $k \geq 2$, the auxiliary channel samples snapshots at increasingly coarser intervals, progressively expanding the model's temporal field of view. In this case, the auxiliary sequence is a subsequence of the original snapshot sequence.

As the stride k widens, the auxiliary path retains fewer snapshots, leading to a progressive loss of fine-grained intermediate transitions. Consequently, the results elucidate a non-monotonic relationship between temporal resolution and model performance, revealing an optimal granularity at $k = 2$. When the stride is minimal (k = 1), the auxiliary channel becomes highly redundant with respect to the main channel, providing limited additional temporal abstraction while still incurring extra computation. Conversely, when k > 2, the auxiliary sequence becomes overly sparse: although it still captures very long-range trends, it loses many medium-range structural changes and event bursts that are crucial for temporal node classification. This undersampling diminishes the complementary information provided by the auxiliary branch, thereby reducing the efficacy of cross-channel fusion. Consequently, a moderate stride (k = 2) achieves the optimal balance between representational granularity and computational efficiency for HASNN.

5.5.2. Effect of neuron type

To further validate the effectiveness of spiking dynamics, we replace the LIF neuron model with conventional neural activation functions, including IF and BLIF. As shown in Table 5, the LIF-based variant consistently outperforms the other two variants, demonstrating the benefit of its biologically inspired decay and reset mechanisms. The IF neuron, a simplified version without the leak term, yields slightly lower results, indicating that the temporal decay in LIF contributes to more effective dynamic encoding. In contrast, the BLIF model, which triggers spikes whenever the membrane potential crosses either a positive or a negative threshold, breaks biological plausibility and significantly alters the spiking statistics. Specifically, in dynamic graphs with highly irregular interaction patterns, this bidirectional firing behavior increases the number of spike events and reduces the sparsity of the spiking representation, making the model more sensitive to minor fluctuations and noise in the input stream. Moreover, the adaptive threshold update in BLIF introduces additional variability in the firing condition, which can destabilize membrane dynamics and make optimization more difficult. Together, these factors lead to less stable and less semantically consistent node representations and thus degraded performance, as reflected by the substantial performance drop in the BLIF setting. This comparison confirms that biologically consistent, sparsity-inducing spike dynamics are critical for robust temporal modeling in dynamic graphs.

5.5.3. Effect of fusion ratio between the two attention channels

As shown in Fig. 5, under the setting where 60% of the data is used for training, we evaluate model performance with different fusion ratios between the two temporal attention branches with and without positional encoding. We vary the weight of the position-encoded branch (w_{pos}) from 0.0 to 1.0, satisfying $w_{pos} + w_{npo} = 1$. The model reaches the best representational performance when $w_{pos} = 0.6$ and $w_{npo} = 0.4$, confirming that this balance between positional precision and global context most effectively captures temporal structure.

Table 5
The selection of neuron types.

Dataset	Metrics	Tr. ratio	LIF	IF	BLIF
DBLP	**Macro-F1**	**40%**	**76.53**	74.19	67.86
		60%	**79.21**	76.11	69.67
		80%	**82.13**	78.69	72.34
	Micro-F1	**40%**	**77.25**	74.65	69.97
		60%	**79.57**	76.64	70.96
		80%	**82.45**	78.99	73.10
Tmall	**Macro-F1**	**40%**	**63.29**	61.23	56.91
		60%	**65.12**	63.72	58.87
		80%	**67.23**	65.50	61.45
	Micro-F1	**40%**	**67.27**	65.43	63.31
		60%	**69.12**	67.80	65.17
		80%	**70.24**	68.98	66.45

6. Spiking activity analysis

6.1. Firing rate analysis

We analyze model firing rates to characterize temporal dynamics and spiking activity over time. Considering HASNN employs both main and auxiliary temporal channels, we extend the definition of firing rate to incorporate the total spike activity across both pathways. Specifically, we compute the firing rate by aggregating the number of spikes and time steps across the two channels, normalized by the number of neurons involved. We define the firing sparsity:

$$firingRate = \frac{\#spikes_{main} + \#spikes_{aux}}{(\#timesteps_{main} + \#timesteps_{aux}) \times \#neurons} \tag{22}$$

We compute the average firing rate across four temporal windows: (0, 0.3T], (0.3T, 0.6T], (0.6T, T], and the overall range (0, T]. As shown in Fig. 6, the DBLP dataset exhibits a significantly lower firing rate in the early stage compared to Tmall and Patent, indicating that its node and edge evolution is relatively limited in the short term, resulting in sparser neural activations. In contrast, Tmall and Patent maintain consistently higher firing rates during the mid-to-late stages, reflecting more active temporal dynamics. In the final stage, the firing rates across all datasets converge, suggesting that as structural evolution accumulates over time, the model tends to increase spiking activity regardless of data type, to preserve richer structural and temporal information.

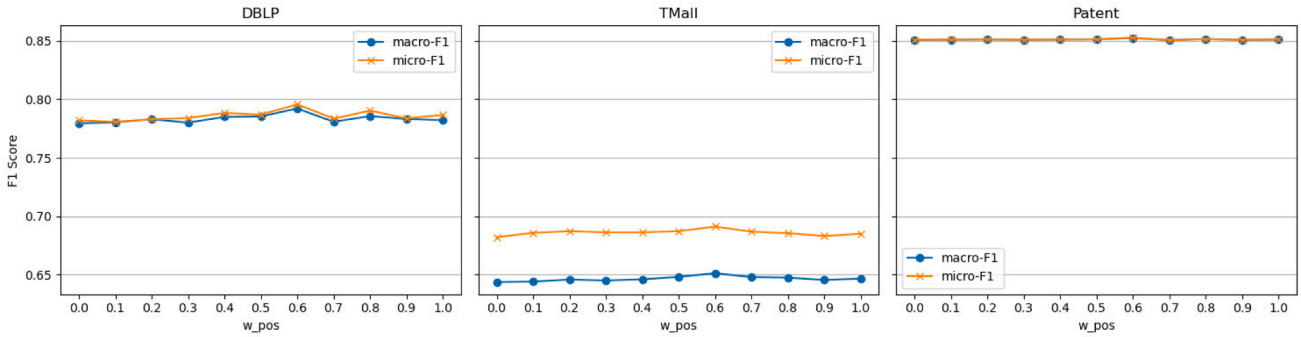

Fig. 5. The fusion ratio between the position-encoded and non-position-encoded attention channels.

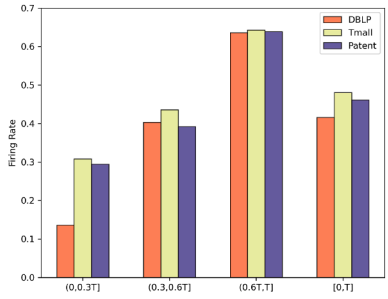

Fig. 6. Average firing rates over time intervals.

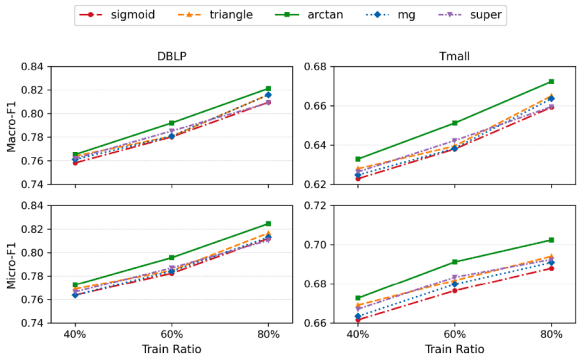

Fig. 7. Performance comparison of HASNN with different spike surrogates.

6.2. Firing mode analysis

To examine the influence of pulse firing patterns on HASNN, we compare several surrogate spike functions used in the LIF neuron, including *sigmoid*, *triangle*, *arctan*, *multi-Gaussian* (mg) and *SuperSpike* (super).

Fig. 7 reports Macro-F1 and Micro-F1 on DBLP and Tmall under different train ratios for the five surrogate functions. Overall, all firing modes yield very similar curves, and the performance consistently improves as the training ratio increases. This indicates that HASNN exhibits strong generalization capabilities and is robust to the specific choice of surrogate gradient, provided that the spike-based gating mechanism is preserved.

At the same time, we observe that the *arctan* surrogate achieves the best accuracy on both datasets under all training ratios. This suggests that a sharper, yet smooth nonlinearity around the firing threshold is slightly more favorable for our spike residual aggregation, but the model still benefits from sparse spike activations under all considered firing modes. Consequently, we adopt arctan as the default function for our main experiments. The stability observed across different functions serves as compelling evidence that the spiking component is a reliable and effective contributor to HASNN, rather than a fragile or hyperparameter-sensitive module.

7. Embedding visualization

To qualitatively assess the discriminative power of the learned node embeddings, we extract the final-layer representations from three representative models: SpikeNet, SiGNN, and the proposed HASNN. For each dataset, we apply t-SNE [35] to project the high-dimensional embeddings of all test nodes into a 2D space, and color each point according to its ground-truth class. We use the same t-SNE setting for all methods on each dataset. The resulting visualizations are shown in Fig. 8.

For SpikeNet (Fig. 8(a)), the clusters corresponding to different classes are relatively entangled and exhibit noticeable overlap, indicating that its embeddings are only weakly separable. SiGNN (Fig. 8(b)) yields more structured patterns, but several classes are still fragmented or partially mixed, especially on the Tmall and Patent datasets.

In contrast, HASNN (Fig. 8(c)) produces clearly separated and compact clusters across all three datasets, with sharper inter-class boundaries and fewer outliers. This improved separability suggests that HASNN learns representations with stronger class-discriminative structure. This visualization aligns well with our quantitative results and suggests that the proposed hierarchical spiking and temporal attention mechanisms enable HASNN to learn more discriminative and semantically coherent node representations than existing SNN-based baselines.

8. Efficiency analysis

8.1. Time complexity

We analyze the time complexity of HASNN based on its multi-hop message passing and dual-path architecture. Let T denote the number of dynamic graph snapshots, $|V|$ the number of nodes per snapshot, K the maximum hop count, S_i the sampled neighborhood size at the i-th hop, and d_h the hidden dimension.

For multi-hop neighborhood sampling and temporal processing, the complexity of feature aggregation is $(T\sum_{i=1}^{K}|V|\prod_{j<i}S_jS_i)$. The core graph aggregation and spike-driven activation operations are executed in both the main and auxiliary channels. The main channel operates over all T steps, while the auxiliary channel is executed approximately every $T/2$ steps, so the total number of SRGA layer invocations is CT, where $1 < C < 2$. When applying two linear projections per node at the k-th hop, the overall cost of spike residual aggregation becomes $O\left(CT|V|\prod_{i=1}^{K}S_i d_h^2\right)$. For the dual-branch temporal attention module (with and without positional encoding), each branch computes a single-head temporal attention score for every node and time step, with complexity $O(T|V|d_h)$ per branch. Hence, temporal attention on one channel

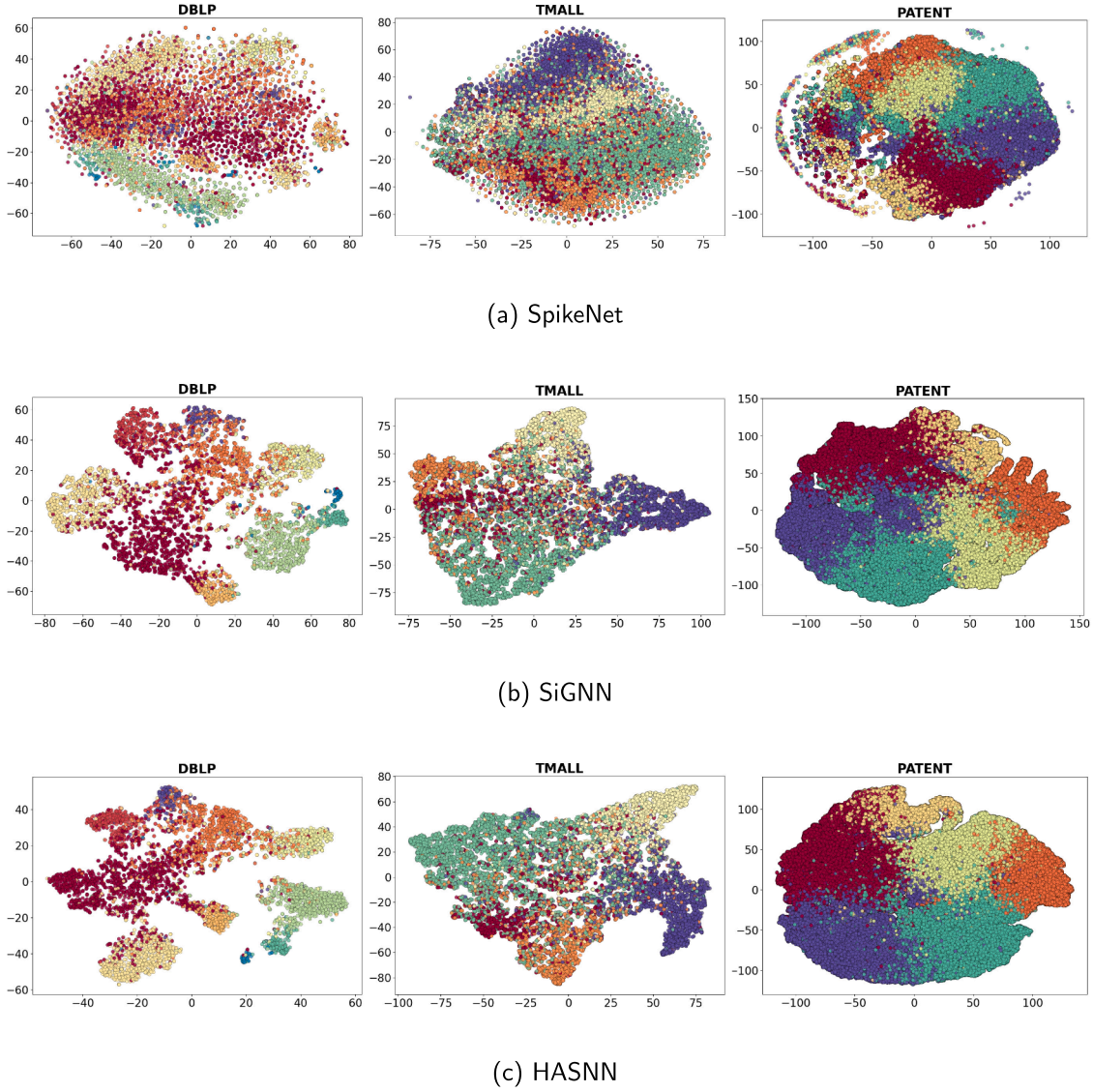

Fig. 8. Visualization of node embeddings on different methods.

costs $O(2T|V|d_h)$ and remains linear in both T and $|V|$, which is dominated by the SRGA term $O\left(CT|V|\prod_{i=1}^{K} S_i d_h^2\right)$. The cross-channel fusion is implemented as lightweight channel-wise attention on temporally aggregated embeddings, with complexity $O(C|V|d_h^2)$.

Combining these components, the overall time complexity of HASNN is determined by the dominant term $O\left(CT|V|\prod_{i=1}^{K} S_i d_h^2\right)$, while the dual-path structure and dual-branch temporal attention only introduce small constant-factor overheads.

Fig. 9 reports the empirical per-epoch training time across all datasets, which is consistent with the above theoretical analysis. SpikeNet achieves the fastest training speed due to its compact single-path architecture. HASNN is slightly slower but remains within the same order of magnitude, and both SiGNN and dense temporal GNNs (TGAT, EvolveGCN) incur significantly higher computational overhead. In addition, Fig. 9 also compares the FLOPs of HASNN, SpikeNet and SiGNN across these datasets. We observe that HASNN consistently requires fewer FLOPs than SiGNN, though moderately more than SpikeNet. This result aligns with the runtime observations of the dual-path and dual-branch design.

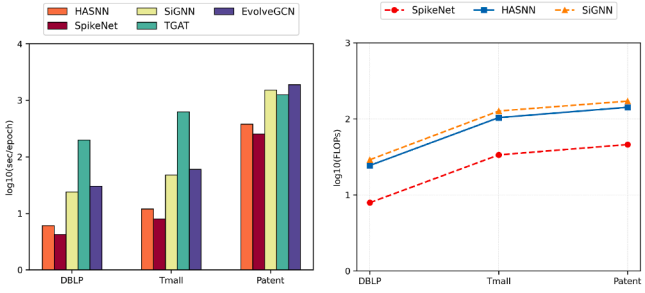

Fig. 9. Training time and FLOPs analysis.

8.2. Parameters and memory

We analyze the space complexity of HASNN from the perspectives of model parameters and GPU memory. The multi-hop SRGA layers and the dual-path structure mainly contribute $O\left(\prod_{i=1}^{K} S_i d_h^2\right)$ parameters per layer, while the temporal attention and channel-wise fusion only

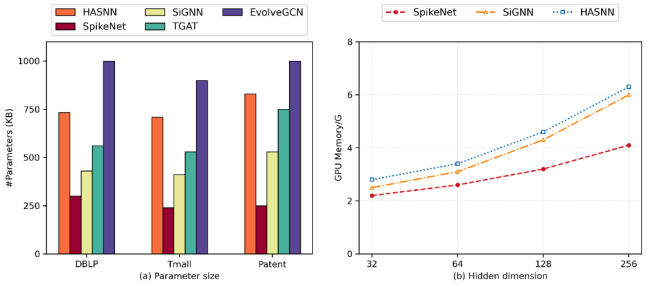

Fig. 10. Comparison of parameters and memory usage.

introduce additional $O(d_h^2)$ and $O(d_h)$ parameters, respectively. Thus, the dual-path SRGA and dual-branch attention increase the parameter count by a modest constant factor but do not change the overall $O(d_h^2)$ scaling.

Fig. 10 reports the parameter sizes and peak GPU memory usage across the three datasets. HASNN has more parameters than SpikeNet, SiGNN and TGAT, but remains clearly smaller than EvolveGCN. Regarding memory consumption, all methods demonstrate linear scaling with respect to the hidden dimension. In this metric, HASNN remains efficient, requiring only slightly more memory than SiGNN.

9. Conclusion

In this paper, we propose HASNN, a novel hierarchical attention spiking neural network for spatio-temporal modeling in dynamic graphs. HASNN focuses on learning expressive and robust node representations by seamlessly integrating biologically inspired spiking neuron dynamics with multi-scale temporal modeling. The core of HASNN is a dual-path architecture that effectively captures both macro- and micro-level temporal evolution patterns within graph structures. Specifically, HASNN incorporates a temporal-aware channel attention mechanism enriched with positional encoding to dynamically assign importance to node representations across time steps, enabling the model to concentrate on critical interaction events. To address the inherent discontinuity in spike-based models, we developed a Spike Residual Graph Aggregation module. This module leverages sparse, event-driven activations from LIF neurons while incorporating cross-layer residual connections to preserve low-level continuous semantics. Comprehensive experiments on multiple benchmark dynamic graph datasets demonstrate that HASNN consistently outperforms competing methods, including both static GNN extensions and other SNN-based approaches, in terms of modeling accuracy and representation quality. Ablation studies further confirm the independent effectiveness of each module and underscore the synergistic benefits of dual-path coordination and spiking-driven temporal learning. Importantly, HASNN exhibits strong modularity and scalability, making it easily adaptable to a wide range of graph neural network architectures. As future work, we plan to extend HASNN to heterogeneous dynamic graphs and real-time graph streams, and to further investigate the theoretical underpinnings and representational capabilities of spiking mechanisms within the dynamic graph learning paradigm.

CRediT authorship contribution statement

Yanglan Gan: Writing – review & editing, Methodology, Investigation, Conceptualization; **Yanzu Dong:** Writing – original draft, Validation, Software, Methodology, Formal analysis, Data curation; **Wenjing Guo:** Writing – review & editing; **Cairong Yan:** Writing – review & editing; **Guobing Zou:** Writing – review & editing, Visualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was sponsored in part by the National Natural Science Foundation of China (62572114, 62272290 and 62477006).

References

[1] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[2] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: a survey, Knowl. Based Syst. 151 (2018) 78–94.
[3] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, D. Yin, Graph neural networks for social recommendation, in: The World Wide Web Conference, 2019, pp. 417–426.
[4] W. Jiang, J. Luo, Graph neural network for traffic forecasting: a survey, Expert Syst. Appl. 207 (2022) 117921.
[5] W. Weng, J. Fan, H. Wu, Y. Hu, H. Tian, F. Zhu, J. Wu, A decomposition dynamic graph convolutional recurrent network for traffic forecasting, Pattern Recognit. 142 (2023) 109670.
[6] J.T. Aparicio, E. Arsenio, F. Santos, R. Henriques, Using dynamic knowledge graphs to detect emerging communities of knowledge (2024).
[7] L. Liao, L. Zheng, F. Chen, J. Shang, X. Li, W. Li, ERD-Net: Modeling entity and relation dynamics for temporal knowledge graph reasoning, Knowl. Based Syst. 317 (2025) 113404.
[8] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).
[9] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, V. Prasanna, Graphsaint: Graph sampling based inductive learning method, (2019). arXiv:1907.04931
[10] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, J. Leskovec, Strategies for pre-training graph neural networks, (2019). arXiv:1905.12265
[11] H. Chi, Y. Lu, Y. Zhu, W. Ke, H. Mao, Dynamic spatiotemporal graph convolutional network collaborative pre-training learning for traffic flow prediction, Knowl. Based Syst. (2025) 114339.
[12] J. Xu, J. Chen, S. You, Z. Xiao, Y. Yang, J. Lu, Robustness of deep learning models on graphs: a survey, AI Open 2 (2021) 69–78.
[13] T. Wu, C. Cui, X. Xian, S. Qiao, C. Wang, L. Yuan, S. Yu, Understanding the robustness of graph neural networks against adversarial attacks, Knowl. Based Syst. (2025) 113714.
[14] B. Li, D. Pi, Learning deep neural networks for node classification, Expert Syst. Appl. 137 (2019) 324–334.
[15] W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao, et al., A comprehensive survey on deep graph representation learning, Neural Netw. 173 (2024) 106207.
[16] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, C. Leiserson, Evolvegcn: evolving graph convolutional networks for dynamic graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, 34, 2020, pp. 5363–5370.
[17] A. Sankar, Y. Wu, L. Gou, W. Zhang, H. Yang, Dynamic graph representation learning via self-attention networks, (2018). arXiv:1812.09430
[18] H.P. Sajjad, A. Docherty, Y. Tyshetskiy, Efficient representation learning using random walks for dynamic graphs, (2019). arXiv:1901.01346
[19] S. Kumar, X. Zhang, J. Leskovec, Predicting dynamic embedding trajectory in temporal interaction networks, in: Proceedings of the 25Th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1269–1278.
[20] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, K. Achan, Inductive representation learning on temporal graphs, (2020). arXiv:2002.07962
[21] S. Molaei, G. Niknam, G.O. Ghosheh, V.K. Chauhan, H. Zare, T. Zhu, S. Pan, D.A. Clifton, Temporal dynamics unleashed: elevating variational graph attention, Knowl. Based Syst. 299 (2024) 112110.
[22] J. Wu, M. Cao, J.C.K. Cheung, W.L. Hamilton, Temp: Temporal message passing for temporal knowledge graph completion, (2020). arXiv:2010.03526
[23] K. Roy, A. Jaiswal, P. Panda, Towards spike-based machine intelligence with neuromorphic computing, Nature 575 (7784) (2019) 607–617.
[24] J. Li, Z. Yu, Z. Zhu, L. Chen, Q. Yu, Z. Zheng, S. Tian, R. Wu, C. Meng, Scaling up dynamic graph representation learning via spiking neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 37, 2023, pp. 8588–8596.
[25] E.O. Neftci, H. Mostafa, F. Zenke, Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks, IEEE Signal Process. Mag. 36 (6) (2019) 51–63.
[26] D. Chen, S. Zheng, M. Xu, Z. Zhu, Y. Zhao, SiGNN: a spike-induced graph neural network for dynamic graph representation learning, Pattern Recognit. 158 (2025) 111026.
[27] W. Nicola, C. Clopath, Supervised learning in spiking neural networks with FORCE training, Nat. Commun. 8 (1) (2017) 2208.
[28] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G.A.F. Guerra, P. Joshi, P. Plank, S.R. Risbud, Advancing neuromorphic computing with loihi: a survey of results and outlook, Proc. IEEE 109 (5) (2021) 911–934.

[29] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, L. Shi, Direct training for spiking neural networks: faster, larger, better, in: Proceedings of the AAAI Conference on Artificial Intelligence, 33, 2019, pp. 1311–1318.
[30] Z. Zhu, J. Peng, J. Li, L. Chen, Q. Yu, S. Luo, Spiking graph convolutional networks, (2022). arXiv:2205.02767
[31] J. Li, H. Zhang, R. Wu, Z. Zhu, B. Wang, C. Meng, Z. Zheng, L. Chen, A graph is worth 1-bit spikes: When graph contrastive learning meets spiking neural networks, (2023). arXiv:2305.19306
[32] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, Y. Tian, Incorporating learnable membrane time constant to enhance learning of spiking neural networks, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 2661–2671.
[33] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[34] A. Grover, J. Leskovec, Node2vec: scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.
[35] M.L. van der, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.