

A Comprehensive Survey of Dynamic Graph Neural Networks: Models, Frameworks, Benchmarks, Experiments and Challenges

ZhengZhao Feng¹, Rui Wang^{1,2,*}, TianXing Wang¹, Mingli Song^{1,3}, Sai Wu^{2,1}, Shuibing He¹

¹ Zhejiang University, Hangzhou, China

² Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security, Hangzhou, China

³ Shanghai Institute for Advanced Study, Zhejiang University, Shanghai, China

{fengzhengzhao,rwang21,tianxingwang,brooksong,wusai,heshuibing}@zju.edu.cn

ABSTRACT

Dynamic Graph Neural Networks (GNNs) combine temporal information with GNNs to capture structural, temporal, and contextual relationships in dynamic graphs simultaneously, leading to enhanced performance in various applications. As the demand for dynamic GNNs continues to grow, numerous models and frameworks have emerged to cater to different application needs. There is a pressing need for a comprehensive survey that evaluates the performance, strengths, and limitations of various approaches in this domain. This paper aims to fill this gap by offering a thorough comparative analysis and experimental evaluation of dynamic GNNs. It covers 81 dynamic GNN models with a novel taxonomy, 12 dynamic GNN training frameworks, and commonly used benchmarks. We also conduct experimental results from testing representative nine dynamic GNN models and three frameworks on six standard graph datasets. Evaluation metrics focus on convergence accuracy, training efficiency, and GPU memory usage, enabling a thorough comparison of performance across various models and frameworks. From the analysis and evaluation results, we identify key challenges and offer principles for future research to enhance the design of models and frameworks in the dynamic GNNs field.

PVLDB Reference Format:

ZhengZhao Feng, Rui Wang, TianXing Wang, Mingli Song, Sai Wu, Shuibing He. A Comprehensive Survey of Dynamic Graph Neural Networks: Models, Frameworks, Benchmarks, Experiments and Challenges. PVLDB, 0(0): XXX-XXX, 2024. doi:XX.XX/XXX.XX

PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at https://github.com/fengwudi/DGNN_model_and_data.

1 INTRODUCTION

Graphs are essential for representing, analyzing, interpreting, and predicting real-world phenomena [104]. Graph neural networks (GNNs), such as GCN [57], GraphSAGE [35] and GAT [115], combine traditional graph computation with deep learning techniques, achieving success in tasks like link prediction [142], node classification [128], and attribute prediction [100]. To enhance the extraction

of real-world dynamic graph insights, dynamic GNNs (DGNNs) like EvolveGCN [87], T-GCN [149], JODIE [62], and TGN [95] integrate temporal information with GNNs to capture structural, temporal, and contextual relationships within dynamic graphs [86]. These dynamic GNN models surpass static GNN models in many tasks with potential applications in social network analysis [19], time series prediction [62], and traffic flow forecasting [149].

Dynamic GNNs or temporal GNNs have gained significant attention in the literature [55]. A plethora of DGNN models and frameworks have been created to address various applications [42, 124, 137, 140, 141, 152], enhance inference accuracy [83, 87, 126, 143], and improve training efficiency [9, 17, 69, 92, 131]. While several surveys on dynamic GNN models exist, they primarily focus on algorithms that fit the encoder-decoder architecture [36]. For instance, [55] review representation learning techniques for dynamic graphs, [105] explores the application of DGNN models in dynamic graph analysis, and [160] proposes a three-stage recursive temporal learning framework based on dynamic graph evolution theory. Furthermore, [51] exclusively concentrates on spatio-temporal graphs, while [76] presents a unique classification approach for DGNN models but with a limited scope. Although these surveys offer valuable insights, they are constrained to a narrow subset of the DGNN development landscape, exhibiting certain limitations:

L1: Outdated coverage of research. The surveys by [55, 105, 160] are somewhat dated, missing out on the numerous new DGNN models that have emerged after their publication. On the other hand, while [51] and [76] provide more current perspectives, the former concentrates solely on spatio-temporal graphs, and the latter examines only a restricted range of DGNN models.

L2: Absence of discussion on DGNN frameworks. Apart from a brief mention in [76] regarding TGL, none of the surveyed works discuss DGNN frameworks. Yet, these frameworks are essential for integrating models, optimizing training, and improving performance and scalability. They serve as a centralized platform for crafting various DGNN architectures, integrating efficient data processing, parallel computation, and tailored optimization algorithms for dynamic graph data. Moreover, DGNN frameworks facilitate advanced functionalities such as distributed training, crucial for handling large-scale datasets and real-world applications.

L3: Oversight of evaluation benchmarks. These surveyed works lack a detailed overview of evaluation benchmarks. While such benchmarks may be known to experienced researchers, newcomers may benefit from a comprehensive explanation. It is essential to introduce evaluation benchmarks thoroughly, including definitions and usage methods, to assist readers in understanding these criteria.

*Corresponding author.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 0, No. 0 ISSN 2150-8097.
doi:XX.XX/XXX.XX

L4: Lack of experimental comparisons. Existing surveyed works serve as foundational overview of DGNN models but lacks experimental comparisons among these models. Such comparisons are vital for grasping performance discrepancies across various models. Challenges arise in unequivocally ranking DGNN models due to differences in datasets and evaluation metrics. Some models excel on specific datasets but may falter on others, lacking scalability. Moreover, variations in experimental settings can yield different results. Therefore, a standardized experimental setup with comprehensive and fair comparisons is necessary to tackle these issues.

L5: Meeting emerging application demands and new challenges. As technology advances, DGNN models have the potential to revolutionize various emerging fields, albeit not yet fully leveraged. Previous research has addressed existing challenges to some extent; however, the emergence of new application demands poses fresh hurdles for Dynamic GNNs to navigate.

In response to the limitations mentioned above, we target to offer a comprehensive and up-to-date overview of dynamic GNNs, encompassing recent research advancements. We introduce new classification methods to adapt to the evolving landscape of dynamic GNN models and explore existing frameworks and evaluation benchmarks. Additionally, we conduct thorough experimental comparisons of prominent DGNN models and frameworks, and examine emerging application demands and challenges in the field of dynamic GNNs. Our main contributions are outlined as follows:

C1: Comprehensive survey and novel taxonomy of DGNN models. Addressing the limitation mentioned in L1, we conducted an extensive survey of 81 recent DGNN models and introduced a novel classification approach (§3). This taxonomy provides unique insights and perspectives in the current research field. By categorizing DGNN models according to their structures, features, and dynamic modeling methods, we facilitate a better understanding and comparison of the strengths and weaknesses of each model.

C2: Overview of existing DGNN frameworks. For limitation L2, we provide a detailed overview of current 12 DGNN frameworks, exploring their features and improvements in model optimization (§4). Our emphasis on the frameworks’ flexibility, scalability, and performance underscores their essential characteristics.

C3: Introduction of evaluation benchmarks for DGNN. To address L3, we present a diverse set of commonly used evaluation graph datasets and metrics for the models discussed (§5). This extensive coverage aims to facilitate comprehensive evaluations and enhance reproducibility in various experiments.

C4: Experimental comparison of selected works. To address L4, we conduct a comprehensive comparison of various DGNN models and frameworks under consistent experimental settings, datasets, and metrics (§6). We evaluate training accuracy, efficiency, and memory usage of these models and frameworks. We also assess multi-GPU scalability within the frameworks.

C5: Analysis of challenges in DGNN. To address L5, we analyze the new challenges encountered by DGNN models and frameworks and suggest potential research directions for practitioners (§7).

2 BACKGROUND

2.1 Applications of Dynamic Graph Learning

2.1.1 Dynamic Graph Scenarios. Real-world graphs exhibit dynamic features and find applications in various domains, including: **Temporal Interaction Graphs in Social Networks:** Temporal interaction graphs capture the evolving relationships and interactions between social network users over time, offering insights into social dynamics and network evolution.

Real-Time Transaction Graphs in E-Commerce: Transaction graphs model the flow of transactions and interactions between users and products in real-time, facilitating fraud detection, recommendation systems, and personalized marketing strategies.

Spatio-Temporal Graphs (STG): These graphs integrate spatial and temporal dimensions, with nodes representing spatial locations and edges denoting spatial relationships. They enable the analysis of movement patterns, traffic flow, and environmental changes, aiding urban planning, transportation management, and environmental monitoring. Temporal information for nodes and edges can enhance the understanding of time-dependent spatial relationships.

Temporal Knowledge Graphs (TKG): Knowledge graphs depict structured information, with entities as nodes and relationships as edges. Adding a temporal dimension in the triplets, TKGs allow to track the changes in entities and relationships over time, essential for applications like online social networks and trend analysis.

Temporal Citation Graphs (TCG): TCGs track the evolution of citations and references in scholarly publications over time, enabling analyses of research trends, influence dynamics, and knowledge dissemination patterns within academic fields.

2.1.2 Learning Tasks in Dynamic Graphs. Dynamic graph learning can aid in various tasks in above application domains, including:

Link Prediction: Involves predicting the likelihood of connections between two nodes in a network that do not yet have edges based on existing network nodes and structure. In dynamic graph learning, link prediction focuses on forecasting the probability of edges appearing between nodes at a specific time.

Node Classification: Entails assigning labels to nodes with unknown labels in a graph by utilizing the connections between nodes and a limited number of labeled nodes. In dynamic graph learning, this task aims to predict the labels of nodes at a given time.

Other Tasks: Apart from the above two common tasks, there are additional specialized tasks, such as temporal node embedding which learns low-dimensional representations of nodes that capture the temporal dynamics and structural changes in the graph, temporal graph embedding which learns representations of the entire graph at different time steps, and event prediction which predicts future events or occurrences in a dynamic graph

2.2 Representation of Dynamic Graphs

2.2.1 Graph Structure Representation. We now discuss the graph structure representation from static graphs to dynamic graphs.

Static Graph Representation: Static graphs are typically represented by nodes, edges, and features: $G = (V, E, X)$, where V is the node set, E is the edge set, and X represents the feature embeddings of the graph. An adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$ is commonly used to depict a static graph. In the matrix, a value of

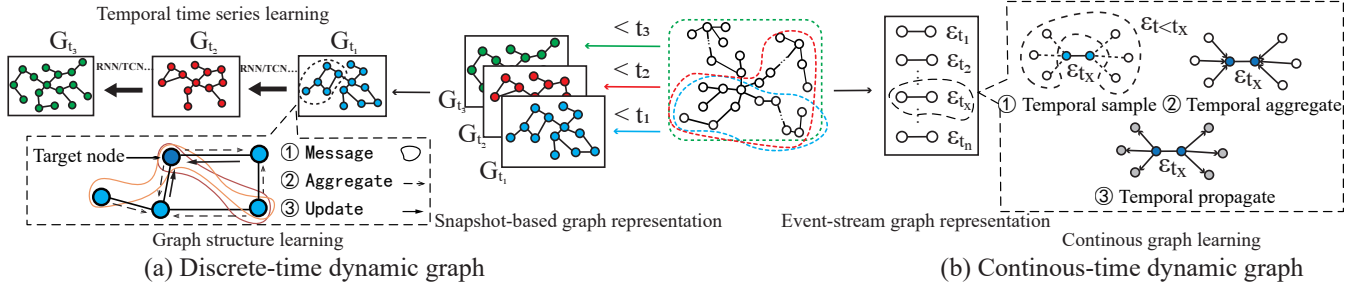


Figure 1: A toy example of dynamic graph representation and learning.

1 at $A[i][j]$ indicates an edge from node i to node j . An edge index $Edge_{index} \in \mathbb{R}^{2 \times |E|}$ is also utilized for graph data, with each column representing an edge in the graph. For example, nodes $Edge_{index}[0][i]$ and $Edge_{index}[1][i]$ define the i -th edge.

Dynamic Graph Representation: Dynamic graphs add a temporal dimension to static graphs. At time t , the graph is represented as $G_t = (V_t, E_t, X_t)$, with V_t, E_t , and X_t representing nodes, edges, and features at that time. Two common approaches to dynamic graph representation are *discrete-time dynamic graphs (DTDG)* and *continuous-time dynamic graphs (CTDG)*. The difference between DTDG and CTDG is illustrated in Figure 1. In DTDG, the timestamp $T_n = [t_1 : t_n]$ is divided into n time intervals, and the dynamic graph is represented as a sequence of graph snapshots within T_n denoted as $G_T = (G_{t_1}, G_{t_2}, \dots, G_{t_n})$. Each G_{t_i} captures the graph structure up to time t_i . On the other hand, in CTDG, graph information is treated as an event stream $G_T = (\epsilon_{t_1}, \epsilon_{t_2}, \dots, \epsilon_{t_n})$. An edge created at time t from node i to node j is represented as $\epsilon_t = (i, j, t)$. CTDG maintains a single graph structure at any given time t by incorporating all event stream data to form G_t .

2.2.2 Temporal Time Series Representation. In the context of dynamic graphs, temporal time series data can be represented using explicit or implicit time methods.

Explicit Time Representation: Explicit time involves including time as a distinct feature in the model for computation. This approach incorporates time series data directly into the model as an input feature, influencing the model’s calculations and enabling it to utilize time information for tasks like prediction or learning.

Implicit Time Representation: On the other hand, implicit time implies that the model comprehends the temporal progression within its structure without explicitly treating time as an input feature. Instead of requiring specific time values, the model learns temporal relationships through the sequential arrangement of data. This allows the model to capture temporal evolution from the data without emphasizing timestamps or explicit time features.

2.3 Learning of Dynamic Graphs

2.3.1 Graph Structure Learning. Diverse methods have been developed to handle various types of graph structure data, ranging from neural networks and attention mechanisms to random walks:

Graph Neural Networks (GNNs): GNNs are neural network models tailored for processing graph data, allowing for the effective capture of intricate relationships within graph structures and enabling node and graph learning and inference. A fundamental aspect of GNN design is the utilization of *message passing mechanisms*, which typically involves three key steps: *message*, *aggregate*, and

update. Specifically, each vertex first collects feature embedding messages from its neighboring vertices, then aggregates the collected feature messages using an aggregation function, and finally updates the vertex’s feature embedding information using neural network model parameters.

Structural Attention: Attention-based graph learning utilizes attention mechanisms in GNNs to prioritize information flow between nodes in a graph. This allows the model to focus on key nodes or edges dynamically, adjusting their importance during message passing and feature aggregation.

Random Walk: Random walk-based graph learning employs random walk algorithms on graphs to capture structural relationships between nodes. In this method, walkers move between nodes based on predefined rules (e.g., edge probabilities) to explore the graph. By conducting multiple random walks and analyzing visited node sequences, we can generate node embeddings that encode the local graph structure and connectivity patterns.

2.3.2 Temporal Time Series Learning. Next, we introduce the learning method for temporal time series information:

Recurrent Neural Networks (RNN): RNN is a classic neural network architecture for processing sequential data like time series and texts. It excels at capturing sequence context and adapting to different sequences lengths. Advanced RNN variations like long short-term memory (LSTM) [43] and gated recurrent unit (GRU) [16] address challenges like gradient vanishing and exploding.

Temporal Point Process (TPP): TPP [18] is a statistical model used to analyze event patterns over time, focusing on event occurrence moments rather than event count or intensity [23, 134]. The conditional intensity function $\lambda(t)$ describes the intensity of future events based on historical event information H_t before time t .

Temporal Convolutional Network (TCN): TCN [7] is a deep learning model for time series data modeling. Unlike RNNs, TCN employs a convolutional neural network structure to capture dependencies in time series, effectively capturing local dependencies within sequential data.

Temporal Attention: Temporal attention is utilized for processing temporal data, enabling models to assign varying importance to information at different time steps when handling sequential data.

Time Encoding: Time encoding involves representing temporal information as features for model training. One common approach is using parameterized Fourier features[94].

2.4 Dynamic Graph Neural Networks

2.4.1 DGNN Models on DTDG and CTDG. In §2.2, we explored how dynamic graphs can be represented using either DTDG or

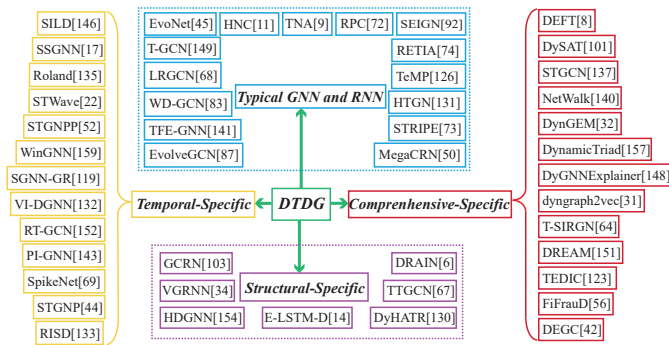


Figure 2: The taxonomy for DTDG in our survey.

CTDG, leading to distinct DGNN models. The training differences for DGNN models in DTDG and CTDG are depicted in Figure 1. While DTDG and CTDG vary in terms of snapshots and event streams, the key contrast lies in the granularity of time representation and learning: coarse-grained time in DTDG and fine-grained time in CTDG. In DTDG, a fixed time interval like a month or a year is often selected to partition the graph into multiple snapshots when edges are added over time. Conversely, CTDG spans an infinite time range, capturing all event streams in a single snapshot graph with precise time ordering and sequential event processing.

3 TAXONOMY OF DYNAMIC GNN MODELS

This section offers a detailed overview of the most recent dynamic GNN models, covering 48 DTDG models and 33 CTDG models. We introduce a new classification method that categorizes these Dynamic GNN models according to their structural features, use of methods, and dynamic modeling techniques.

3.1 Discrete-Time Dynamic Graph Models

DTDG models are designed for analyzing discrete-time dynamic graphs. These models can be classified into four groups based on their methods for handling graph structure and temporal data, as illustrated in Figure 2. **Typical GNN and RNN models** involve using GNNs for processing graph structure data and RNNs for temporal time series data. Other DTDG models fall into the categories of **structural-specific**, **temporal-specific**, and **comprehensive-specific models**, depending on their reliance on non-GNN methods for graph structure processing, non-RNN methods for temporal data processing, or neither GNN nor RNN, respectively.

3.1.1 Typical GNN and RNN Models. The conventional approach of utilizing GNN for processing graph structures and RNN for handling temporal information is considered the most intuitive and prevalent method in DGNN modeling.

Enhancing Accuracy. WD-GCN [83] was among the first algorithms to combine GNN and RNN for dynamic graph processing, merging GCN [57] and LSTM [43] models. EvolveGCN [87] updates the weight matrix of GCN models across different time frames using RNN. TeMP [126] incorporates temporal message passing for improved predictions, blending the structural aspects of Relational GCN and GRU [16]. EvoNet [45] integrates graph-level propagation with GNN and RNN components to comprehensively capture

temporal graph information. RETIA [74] considers adjacent relationships in addition to aggregating neighboring entities, leading to a more holistic learning approach. Incorporating GNN, GRU, and novel units RCU and PCU, RPC [72] explores relational correlations and periodic patterns, enhancing the model’s expressiveness.

Improving Training Efficiency and Scalability. TNA [9] enhances training stability beyond traditional GCN and GRU structures. HTGN [131] maps the temporal graph onto hyperbolic space and introduces specific modules to constrain the model, ensuring stability and generalization of embeddings. SEIGN [92] utilizes a three-part structure involving GCN-like message passing, GRU parameter adjustments over time, and a self-attention mechanism inspired by transformers [114] for learning the final representation. This strategy enhances scalability and efficiency, enabling effective training on large-scale graphs.

Specialized Applications. Several typical models are tailored for specific application or domains. LRGCN [68] incorporates LSTM [43] based on Relational R-GCN [102] for path fault prediction. T-GCN [149] is designed for traffic prediction, combining GCN and GRU. MegaCRN [50] introduces the GCRN unit for processing spatio-temporal graphs, utilizing a hybrid architecture of GCN and GRU. TFE-GNN [141] processes graph structure using multi-layer GNNs. HNC [11] manages satellite communication status within a satellite network, aiding in the development of a global satellite joint program. STRIPE [73] is the first to incorporate memory networks for anomaly detection in spatio-temporal graphs.

3.1.2 Structural-Specific Models. This category of DTDG models utilizes non-GNN methods for processing graph structural data and still uses RNN for temporal time series data processing.

Enhancing Accuracy. GCRN [103] combines GCN and RNN to predict dynamic data using both spatial and temporal information. VGRNN [34] builds upon GCRN by transitioning to GRNN and introducing the dynamic graph auto-encoder model. E-LSTM-D [14] is the first to utilize LSTM and an encoder-decoder architecture for link prediction in dynamic networks. TTGCN [67] introduces a dynamic graph representation learning method based on k-truss decomposition, effectively capturing multi-scale topological structure information in graphs.

Improving Training Efficiency. DRAIN [6] has constructed a recurrent graph generation scenario to represent a dynamic GNN that learns across different time points. It captures the time drift of model parameters and data distributions, and can predict future models in the absence of future data.

Specialized Applications. Some structural-specific models are tailored for managing heterogeneous graphs. HDGNN [154] employs multi-head attention and random walk techniques for processing structural information, whereas DyHATR [130] integrates a hierarchical attention mechanism to learn heterogeneous information.

3.1.3 Temporal-Specific Models. This category of DTDG models employs GNN for processing graph structural data and non-RNN methods for temporal time series data processing.

Enhancing Accuracy. SGNN-GR [119] utilizes generative adversarial networks (GAN) to generate synthetic time information, addressing catastrophic forgetting issues without additional data storage. Paired with GraphSAGE [35], it establishes a framework for DGNN. PI-GNN [143] employs parameter isolation for dynamic

graphs to capture emerging patterns without compromising older ones. STGNP [44] introduces the spatio-temporal graph neural process, incorporating neural processes [29] for modeling spatio-temporal graphs. SILD [146] firstly explores distribution shifts in the spectral domain of dynamic graphs.

Improving Training Efficiency and Scalability. ROLAND [135] introduces embedding update modules to handle adjacent time snapshots, enabling the extension of static graphs to dynamic graphs. WinGNN [159] replaces time encoders with random sliding windows and adaptive gradients, effectively reducing the number of parameters. SSGNN [17] utilizes echo state networks (ESN) [47] to enhance scalability for large networks. SpikeNet [69] replaces RNN functionality with spiking neural networks (SNN) [30], reducing model and computational complexity through the leaky integrate-and-fire (LIF) model of SNN.

Specialized Applications. RT-GCN [152] predicts stocks using relationship and time convolution, employing GCN for relationship convolution and TCN for time convolution. RISD [133] is designed for heterogeneous graphs, utilizing heterogeneous GCN to learn graph representations and construct sampling graphs. STWave [22] is a framework for traffic prediction, incorporating GAT for spatial dimensions and a transformer for temporal dimensions to identify spatio-temporal correlations. STGNPP [52] combines GCN and transformer for predicting traffic congestion time using neural process priors (NPP). VI-DGNN [132] presents a model tailored for trajectory prediction tasks in the intelligent transportation domain.

3.1.4 Comprehensive-Specific Models. The final category comprises models that do not fall into the previous classifications, which neither use GNN nor RNN for dynamic graph processing.

Enhancing Accuracy. Dyngraph2vec [31] is an early Dynamic GNN model capable of capturing temporal dynamics. DynGEM [32] initializes snapshot embeddings from previous time steps before gradient training, avoiding learning from scratch. DySAT [101] incorporates an attention mechanism to learn structural and temporal information efficiently compared to RNN-based solutions.

Improving Scalability. T-SIRGN [64], inspired by SIR-GN [54], extends the capabilities of existing approaches to handle efficiency and scalability issues in dynamic graphs.

Path-based Methods. DynamicTriad [157] utilizes triples for link prediction to capture network dynamics. Netwalk [140] reconstructs node representations and groups embeddings along walking paths using a deep auto-encoder model.

Capturing Global Information. DEFT [8] employs learning spectral wavelets [37] to capture global features, effectively learning dynamic evolution graph representations. DREAM [151] captures long-term evolution through a temporal self-attention network. EA-GLE [139] utilizes the *modeling-infering-discriminating-generalizing* paradigm to enhance extrapolation capabilities in the future.

Specialized Applications. STGCN [137] focuses on spatio-temporal correlations for long-term traffic prediction tasks. TEDIC [123] models information flow using enhanced graph convolution and extracts fine-grained patterns over time for dynamic social interaction pattern extraction. DEGC [42] addresses recommendation system challenges with an isolation-based approach to handle obsolescence. FiFraud [56] is an unsupervised and scalable method for detecting suspicious traders and behavioral patterns efficiently.

DyGNNExplainer [148] introduces a causality-inspired generative model based on structural causal models to explore the philosophy of DyGNN prediction by identifying trivial, static, and dynamic causal relationships.

3.2 Continuous-Time Dynamic Graph Models

In the realm of CTDG models, notable contributions are categorized into **TPP-based models**, **memory-based models**, and **random-walk-based models** based on the methods they employ to analyze temporal events. For models that do not align with these categories, we categorize them into **instant node update model**, **neighbor aggregation-update model**, and **aggregation-update-propagation model** according to their methods for updating nodes using event streams. The distinctions between these four node update methods are depicted in Figure 3. The taxonomy of CTDG models is shown in Figure 4.

3.2.1 TPP-Based Models. TPP is a valuable tool for modeling temporal dynamics, as discussed in §2.3. Several works [40, 59, 77, 112, 113, 150] utilize TPP in their frameworks. Know-E [112] is pioneering in employing TPP for CTDG and capable of predicting potential event occurrence times. It updates node embeddings based on incoming events and recent relationship and time series. DyREP [113] divides model updates into three segments and incorporates TPP to calculate time attention for neighboring nodes. M²DNE [77] introduces macro constraints to scale the network through equations, complemented by micro-level time attention aggregation. GHN [40] utilizes the Hawkes process to capture entity interaction time dynamics in TKGs and introduces a novel time knowledge graph connection prediction ranking metric. LDG [59] enhances previous approaches by addressing issues related to long-term edge information quality. DynShare [150] designs an interval-aware personalized projection operator using TPP to enable diverse user mode selections within the same time interval. While these algorithms all leverage TPP, variations exist in their implementations. For instance, DyREP, GHN, LDG, and DynShare directly incorporate TPP in the entire forward propagation calculation, whereas Know-E and M²DNE use TPP for objective function computations.

3.2.2 Memory-Based Models. A part of CTDG models employs a memory mechanism to retain historical information for facilitating embedded updates. TGN [95] leverages memory to store historical node data, enabling updates upon the arrival of event streams. APAN [121] separates model reasoning from graph computation

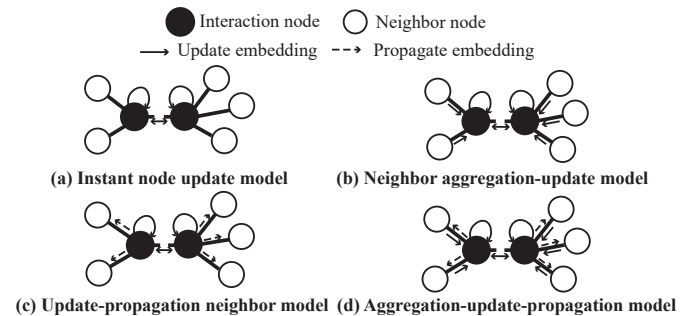


Figure 3: Node update methods based on event streams.

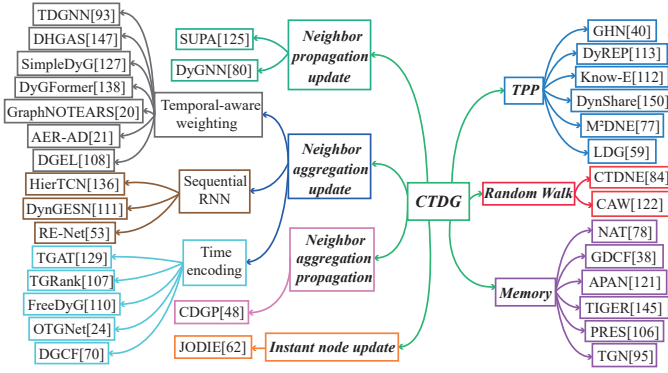


Figure 4: The taxonomy for CTDG in our survey.

through an asynchronous processing approach, employing a mailbox mechanism to propagate information to neighbors’ mailboxes post-event update, eliminating the need for neighbor aggregation during updates. NAT [78] introduces a unique dictionary-based neighborhood representation and N-cache data structure to enable parallel access and updates of these dictionary representations on GPUs. GDGF [38] focuses on crowd flow modeling and employs a memory module to enhance update speed. TIGER [145] introduces a dual memory module to address the limitations of TGN. PRES [106] utilizes gaussian mixture models for correction and predicts new memory based on historical time series gradients.

3.2.3 Random-Walk-Based Models. Random walk has proven to be effective in graph learning, as discussed in §2.3. CTDNE [84] is a pioneering algorithm in dynamic graph embedding that incorporates time embeddings into network embeddings. This model introduces a comprehensive framework that integrates time dependencies into node embeddings and employs a depth map model through random walks. CAW [122] represents a temporal network using causal anonymous random walks extracted from temporal random walk, and adopts an anonymization strategy that keep the method inductive and establishes correlation between motifs.

3.2.4 Instant Node Update Models. These models update the node embedding based solely on the current event and directly involved nodes, without considering the influence of neighbors, as depicted in Figure 3(a). For instance, JODIE [62] uses an embedded projection module to predict the future node embedding trajectory. It partitions the embedding into dynamic and static components to capture time-varying and time-invariant features, respectively.

3.2.5 Neighbor Aggregation-Update Models. These models leverage a combination of event data, historical node information, and neighbor information to update node embeddings, as shown in Figure 3(b). This approach is frequently employed in CTDG models and represents a conventional method for dynamic graph embedding. Within this category, classification can be further refined based on the methods used to integrate time information for graph updates: **Temporal-Aware Weighting Models.** TDGNN [93] introduces a dynamic network representation learning framework that effectively captures node representations by incorporating node characteristics and edge time information using the TDAgg aggregation function. DGELL [108] outlines three key processes (inherent interaction potential, time attenuation neighbor enhancement, and

symbiotic local learning) to comprehensively update dynamic node embeddings with rich graph information. AER-AD [21] focuses on inductive anomaly detection in attribute and non-attribute dynamic graphs, utilizing anonymous edge representation for detecting anomalies in dynamic bipartite graphs in an inductive setting. GraphNOTEARS [20] was the first to study the learning problem of directed acyclic graphs and develop a score-based learning method. SimpleDyG [127] proposes a novel strategy that maps dynamic graphs to sequences to enhance scalability. DyGFormer [138] presents a transformer-based dynamic graph learning architecture that effectively captures node correlations and long-term temporal dependencies. DHGAS [147] introduces the first dynamic heterogeneous graph neural architecture search method, featuring a unified dynamic heterogeneous graph attention (DHGA) framework that allows each node to focus on its heterogeneity and dynamic neighbors simultaneously.

Sequential RNN Models. RE-Net [53] effectively models time, relationships, and interactions between nodes by utilizing RNN to capture the dynamics of time and relationships. Its neighborhood aggregation module combines information from neighboring nodes to handle multiple concurrent interactions at the same timestamp. HierTCN [136] employs RNN at the high-level to learn long-term interests, while TCN [7] is used at the low-level to predict the next interaction based on long-term interests and short-term interactions. DynGESN [111] introduces echo state networks (ESN) [47] for dynamic graph modeling.

Time Encoding Models. Models in this category incorporate time information as part of the embedding to influence the calculation process. TGAT [129] uses self-attention and time encoding to create the TGAT layer, achieving similar processing capabilities as GAT on static graphs by stacking TGAT layers. DGCF [70] effectively models dynamic user-project relationships, capturing both collaborative and sequential connections. OTGNet [24] extends TGAT’s application to open graphs, enabling handling of open-time dynamic graphs. TGRank [107] boosts the model’s link prediction expressiveness, allowing for prediction of crucial structural information. FreeDyG [110] introduces a node interaction frequency encoding module that explicitly captures common neighbor proportions and node pair interaction frequencies to address the *shift* phenomenon.

3.2.6 Update-Propagation Neighbor Models. These models only utilize event and historical node information to update, embed, and propagate data to neighbors, as depicted in Figure 3(c). In DyGNN [80], the nodes engaged in the event are initially updated using LSTM, followed by updating the neighbors of these nodes with event information. SUPA [125] generates a sample graph for the event-involved nodes, updates them with event flow data, and propagates the updated information throughout the sample graph.

3.2.7 Aggregation-Update-Propagation Models. These models rely on historical neighbor information for embedding updates, followed by propagating the updated information to neighbors, as illustrated in Figure 3(d). An example is CDGP [48], a popularity prediction model on community dynamic graphs. CDGP identifies the community for the event, aggregates nodes within the community for embedding updates, and extends the community’s influence to other nodes within the same community.

Table 1: The comparison between the existing dynamic GNN training frameworks

Systems \ Features	Universality		Expandability			Supported functionalities		
	DTDG	CTDG	Single-machine single-GPU	Single-machine multi-GPU	Multi-machine multi-GPU	Temporal neighbor storage	Feature extraction optimizing	Temporal parallel training
PyGT [99]	✓	✗	✓	✗	✗	Store snapshots	✗	✗
ESDG [12]	✓	✗	✓	✓	✓	Store snapshots	✗	Snapshot parallelism
PiPAD [118]	✓	✗	✓	✓	✗	Store snapshots	Extract common snapshots	Snapshot parallelism
DynaGraph [33]	✓	✗	✓	✓	✓	Store snapshots	✗	Snapshot partition parallel
BLAD [26]	✓	✗	✓	✓	✓	Store snapshots	✗	Operator parallelism
TGL [155]	✓	✓	✓	✓	✗	Neighbor static sorting	✗	Small-batch parallelism
DistTGL [156]	✓	✓	✓	✓	✓	Neighbor static sorting	✗	Memory parallelism
NeutronStream [13]	✗	✓	✓	✗	✗	Store event stream	✗	Event group parallelism
SPEED [15]	✗	✓	✓	✓	✗	Store static graph	✗	Subgraph partition parallel
DyGLib [138]	✗	✓	✓	✗	✗	Store event stream	✗	✗
Zebra [71]	✗	✓	✓	✗	✗	Store event stream	✗	✗
GNNFlow [153]	✗	✓	✓	✓	✓	Store event stream	Vectorized cache	✗

4 DYNAMIC GNN TRAINING FRAMEWORKS

In this section, we offer a thorough survey of the latest dynamic GNN frameworks, encompassing 5 discrete-time dynamic graph (DTDG) frameworks and 7 continuous-time dynamic graph (CTDG) frameworks. We start by outlining the requirements for dynamic GNN frameworks, then delve into the existing DTDG frameworks and CTDG frameworks, respectively.

4.1 Needs for DGNN Frameworks.

While research on dynamic GNN models has advanced significantly, the focus has been primarily on enhancing model training accuracy and expanding application domains. The high computational complexity of these models often results in low training efficiency and limited scalability on large graphs, making them suitable mainly for small graphs within thousands of vertices and tens of thousands of edges. This limitation hinders their practical application. Hence, there is a critical requirement for a framework that enhances the training efficiency and scalability of dynamic GNNs on large-scale dynamic graphs by optimizing system execution efficiency.

Recently, specialized GNN training frameworks like PyG [25] and DGL [120] have been developed to simplify programming and enhance training efficiency for various GNN models. These frameworks introduce optimization strategies on top of fundamental training modules. For instance, NeuGraph [79] and Roc [49] optimize vertex access and load balancing through improved graph partitioning strategies. AliGraph [158] and DUCATI [144] reduce data transmission by caching embeddings and intermediate training results. P3 [27] and PipeGCN [117] employ parallel training strategies to accelerate GNN training. However, these frameworks primarily cater to training static graphs and lack essential mechanisms for updating graph structures and modeling temporal information in dynamic GNN models. Key modules like dynamic graph loading, temporal neighbor sampling, and temporal message passing are missing. The temporal dependency of dynamic graphs also complicates parallel computing. Therefore, users must develop these modules themselves to construct dynamic GNN models and carefully handle dynamic graphs to ensure the temporal consistency of data training, resulting in heavy development costs for users.

In the past two years, several dynamic GNN training frameworks have emerged, aiming to better support the training of dynamic GNN models, including both discrete-time and continuous-time dynamic graph models. However, due to the differences in training processes between these two types of models, most frameworks are optimized for only one type of models. We conducted a study on the 12 existing dynamic GNN training frameworks and compiled a detailed comparison of their universality, expandability, and supported functionalities, as outlined in Table 1.

4.2 Discrete-Time DGNN Frameworks

Discrete-time DGNNs typically treat the dynamic graph as a sequence of graph snapshots. PyGT [99], an extension of PyG, introduces three snapshot-based dynamic graph types and provides a unified data loader for each type of dynamic graph to support for training DTDG models. However, it is limited to full-batch training on single GPU for small-scale graphs. ESDG [12] utilizes snapshot partitioning to distribute multiple graph snapshots to different devices for parallel GNN training, and after finishing the GNN computation, redistributes them to colocate the same vertices from different snapshots on the same device for parallel RNN computation. PiPAD [118], based on the similarity of the structures between adjacent graph snapshots, reduces data transmission by transmitting the common parts of a set of consecutive graph snapshots and the individual parts of each graph snapshot, ensuring that these snapshots can be computed in parallel on GPU. DynaGraph [33] proposes a time fusion mechanism to concatenate node features of multiple snapshots in a single graph convolution operation to fully utilize the GPU resource. BLAD [26] explores fine-grained operator-level parallelism to fully leverage GPU resources by simultaneously executing memory-intensive graph operators and compute-intensive neural network operators on a single GPU.

4.3 Continuous-Time DGNN Frameworks

Traditional static graph storage formats like compressed sparse row (CSR) formats are not well-suited for dynamic graphs as they struggle to efficiently handle edge and vertex insertions and deletions. Dynamic graph storage also needs to handle temporal graph

sampling effectively, considering only edges up to the current timestamp. TGL [155] addresses these challenges by introducing a framework that abstracts key components for training dynamic GNN models, including temporal sampler, temporal message mailbox, vertex history memory, memory updater, and message passing engine. It also utilizes sorted static neighbor storage format T-CSR for dynamic graph storage, and parallel samplers and random block scheduling techniques for enhancing training efficiency. GNNFlow [153] proposes a time-indexed block-based data structure for dynamic graph storage to facilitate edge sampling and updates. It also implements caching mechanisms by specifically caching frequently used edge features to optimize CPU-GPU data transfers performance.

CTDG models view dynamic graphs as sequences of events with temporal dependencies, making parallelization challenging. TGL uses mini-batch parallelism but overlooks dependencies within individual mini-batches, potentially impact training accuracy. Neutron-Stream [13] employs an adaptive sliding window training approach to capture temporal dependencies and identify parallelizable event groups while maintaining temporal order. DistTGL [156] extends TGL by introducing distributed training methods, such as epoch parallelism and memory parallelism, to enhance scalability across multiple GPUs. SPEED [15] customizes partitioning strategies for parallel training to ensure load balance and minimize replication factors while preserving temporal information. DyGLib [138] offers a comprehensive library for dynamic graph learning, featuring standardized training processes, scalable coding interfaces, and thorough evaluation protocols. Zebra [71] introduces T-PPR by combining random walks with dynamic GNNs to quickly generate node embeddings through top-k T-PPR queries.

5 DYNAMIC GNN BENCHMARKS

5.1 Datasets

Table 2 shows a summary of commonly used graph datasets in dynamic GNN evaluation. It includes details on application domains, node and edge counts and dimensions, timestamp ranges, label information, and dataset sizes. The datasets highlighted in bold have been chosen for comparison evaluation (§6). Subsequently, we provide a brief introduction to these datasets.

5.1.1 Social Networks. **Reddit** [62] consists of one month of user posts on subreddits, with users and subreddits as nodes and timestamped posting requests as links. **DGraphFin** [46] is the real social network in financial industry provided by Finvolution Group, with nodes representing Finvolution users and edges indicating emergency contacts. **Enron** [58] is an mail dataset capturing interactions among Enron Inc. employees, where communication links denote email exchanges among core employees. **Facebook** [116] is a user interaction network where nodes represent users and edges signify interactions between users. **Social Evolution** [82] comes from the MIT Human Dynamics Lab focusing on social relationship evolution and the degree of closeness between individuals. **UCI** [85] is an online community of students from the University of California, Irvine, with links representing messages exchanged between users.

5.1.2 Interaction Networks. **Wikipedia** [62] is a bipartite interaction graph capturing user edits on Wikipedia pages over a month, with nodes representing users and pages, and links representing

Table 2: Summary of common graph datasets, where $|V|$ and $|E|$ stand for the number of nodes and edges, $Range(t)$ denotes the timestamp range, and $Sizes$ specifies the total file sizes.

Datasets	$ V $	$ E $	$Range(t)$	$Sizes$
Reddit [89]	10984	672447	0~2678390	2.2GB
DGraphFin [3]	4889537	4300999	1~821	649MB
Enron [90]	184	125,235	0~113740399	35MB
Facebook [97]	63731	1269502	1157454929~1232576125	26MB
Social Evolution [81]	74	2,099,520	1188972131~1247740843	148MB
UCI [63]	899	33720	1084560796~1098772901	668KB
Wikipedia [89]	9227	157474	0~2678373	534MB
MOOC [89]	7144	411749	0~2572086	40MB
ML25M [4]	221588	25000095	789652009~1574327703	647MB
LastFM [89]	1980	1293103	0~137107267	37MB
FB-FORUM [98]	899	33720	1084585996~1098798101	612KB
DBLP [1]	28,086	162,451	1~27	375MB
Yelp [5]	2138275	6990280	1108495402~1642592925	5.0GB
ICEWS05-15 [75]	10,094	461,329	1104537600~1451520000	30MB
ICEWS14 [75]	6,869	96,730	1388534400~1419984000	6MB
ICEWS18 [10]	23,033	741820	1514764800~1546214400	184MB
GDELT [2]	16682	191290882	0~175283	82.3GB
Bitcoin-OTC [88]	5881	35592	1289241942~1453684324	988KB
Bitcoin-Alpha [88]	3782	24186	1289192400~1453684324	492KB
AS-733 [88]	7716	11965533	939340800~946771200	115MB
Flights [90]	13169	1927145	0~121	32MB

editing behaviors. **MOOC** [62] records student behaviors in Massive Open Online Courses (MOOCs), such as watching videos and submitting answers. **ML25M** [41] contains user ratings of movies, reflecting their preferences and levels of interest in various movies. **LastFM** [62] provides information on user-listened songs over a month. **FB-FORUM** [96] is the Facebook-like forum network from the same online community as the social network dataset, focusing on user activities in the forum. **DBLP** [109] captures academic collaboration network dataset with nodes representing authors and edges denoting co-authored papers. **Yelp** [101] is a large business review website where users can upload comments to review businesses and discover interested ones based on others' comments.

5.1.3 Event Networks. **ICEWS** [28, 39] and **GDELT** [65] are event networks where nodes represent actors and temporal edges represent point-time events derived from news and articles, such as "yield," "make public statement," "threaten," and more, providing a dynamic view of real-world interactions and occurrences.

5.1.4 Trade Networks. **Bitcoin-OTC** [60] and **Bitcoin-Alpha** [61] are who-trusts-whom networks of bitcoin users trading on the bitcoin-otc platform and btc-alpha platform, respectively.

5.1.5 Traffic Networks. **AS-733** [66] is a communication network composed of routers used to exchange traffic with peers. **Flights** [91] is a transportation map where nodes represent airports and edges represent flights between these airports.

5.2 Metrics

Here is a brief overview of commonly used evaluation metrics to showcase the performance of different dynamic GNN models and frameworks. For detailed calculation equations, refer to Table 3.

5.2.1 Binary Classification Performance. The confusion matrix for binary classification tasks is shown in Table 4, where rows represent predicted classes and columns represent actual classes. **Precision** measures the proportion of correctly predicted positive samples

Table 3: Common evaluation metrics in dynamic GNNs.

Applications	Metrics	Equations
Binary classification	Precision	$\frac{TP}{TP+FP}$
	Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
	Recall	$\frac{TP}{TP+FN}$
	F1 score	$\frac{2 \times Precision \times Recall}{Precision + Recall}$
	AUC	$TPR = \frac{TP}{TP+FN}, FPR = \frac{FP}{FP+TN}$
Multi classification	Micro-F1	(1) $Precision_{micro} = \frac{\sum_i TP_i}{\sum_i TP_i + FP_i}$ (2) $Recall_{micro} = \frac{\sum_i TP_i}{\sum_i TP_i + FN_i}$ $\frac{2 \times Precision_{micro} \times Recall_{micro}}{Precision_{micro} + Recall_{micro}}$
	Macro-F1	(1) $Precision_{macro} = \frac{1}{N} \sum_i \frac{TP_i}{TP_i + FP_i}$ (2) $Recall_{macro} = \frac{1}{N} \sum_i \frac{TP_i}{TP_i + FN_i}$ $\frac{2 \times Precision_{macro} \times Recall_{macro}}{Precision_{macro} + Recall_{macro}}$
Recommendation system	Precision@K	$\frac{TP@K}{TP@K+FP@K}$
	Recall@K	$\frac{TP@K}{TP@K+FN@K}$
	MR	$\frac{1}{ S } \sum_{i=1}^{ S } rank_i$
	MRR	$\frac{1}{ S } \sum_{i=1}^{ S } \frac{1}{rank_i}$
	HITS@K	$\frac{1}{ S } \sum_{i=1}^{ S } IF(rank_i \leq k)$
Regression Model Performance	RMSE	$\sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}}$
	MAE	$\frac{\sum_{i=1}^n y_i - \hat{y}_i }{n}$
	MAPE	$\frac{\sum_{i=1}^n \frac{ y_i - \hat{y}_i }{y_i}}{n}$

Table 4: Confusion matrix in binary classification tasks.

		Prediction	
		Positive	Negative
Actual	True	True Positive(TP)	False Negative(FN)
	False	False Positive(FP)	True Negative(TN)

among all predicted positives. **Accuracy** indicates the proportion of correctly predicted samples out of all samples. **Recall** assesses the proportion of actual positive samples correctly predicted. **F1 Score** is the harmonic mean of precision and recall, offering a balanced measure of classification performance for both positive and negative cases. **AUC** (Area under the curve) typically denotes the area under the ROC curve, a tool for assessing binary classification models using True Positive Rate (TPR) and False Positive Rate (FPR).

5.2.2 Multi Classification Performance. In multi-class classification tasks, key performance metrics include: **Micro-F1 Score** computes the F1 score considering total true positives, false positives, and false negatives across all categories. **Macro-F1 Score** computes the average of F1 scores for individual categories. **Micro-AUC** is calculated by micro-averaging true positive rates and false positive rates for all categories. **Macro-AUC** is obtained by macro-averaging AUC values for each category.

5.2.3 Recommendation System Performance. In recommendation system tasks, **Precision@K** measures the proportion of the top K recommended items to the total number of recommended items. **Recall@K** measures the proportion of the top k recommended

items to the total number of relevant items. **MR** (Mean Rank) measures the average recommended ranking $rank_i$ of all users S . **MRR** (Mean Reciprocal Rank) measures the average reciprocal of the recommended ranking of all users. **HITS@K** measures the accuracy of the top K recommendation results by assessing how many of them align with users’ genuine interests. Function $IF(\cdot)$ outputs 1 if the condition is true and 0 otherwise.

5.2.4 Regression Model Performance. In recommendation systems, **RMSE** (Root Mean Squared Error) quantifies the difference between predicted and actual values, while **MAE** (Mean Absolute Error) calculates the average error between predicted and actual values. **MAPE** (Mean Absolute Percentage Error) measures the average percentage difference between predicted and actual values.

5.2.5 Efficiency and Scalability. **Throughput** measures the number of tasks or data processed within a specific time frame. In deep learning, it typically refers to the number of samples that the model can handle per unit time, providing insights into processing speed. **Training time** refers to the duration time the model spends in the training phase. **GPU memory usage** indicates the amount of memory occupied during training or inference on a GPU. Efficient memory management is crucial for optimizing model performance and scalability. **Parameter size** quantifies the total number of learnable parameters in the model. The parameter size directly impacts the model’s complexity, storage requirements, and computational efficiency, influencing scalability and performance.

6 EXPERIMENTAL COMPARISON

We thoroughly evaluate various dynamic GNN models and frameworks in terms of training accuracy, efficiency, and memory usage. We first compare several classic DTDG and CTDG GNN models (§6.2), then compare models implemented on optimized frameworks (§6.3), analyze multi-GPU scalability (§6.4).

6.1 Experiment Setting

Test Setup: We perform our experiments on two Ubuntu 22.04.3 LTS machines, each featuring an Intel Xeon Gold 6342 CPU @ 2.80GHz and four Nvidia A40 48GB GPUs. These machines are equipped with 1TB of memory and 40TB of disk space.

Baseline Models and Frameworks: We compare six CTDG GNN models (JODIE, TGAT, TGN, APAN, CAW, DyREP) and three DTDG GNN models (EvolveGCN, Roland, DySAT), along with three dynamic GNN frameworks (TGL, DistTGL, SPEED). For JODIE and DyREP models, we use the implementation provided by TGN. We utilize the most efficient version, TGN-attn, for the TGN model. EvolveGCN have two versions EvolveGCN-H and EvolveGCN-O, using GRU and LSTM as components of learning time, respectively. All CTDG models are set by default with 1 GNN layer and a batch size of 1000, and follow the same early stopping condition (3 epochs) as AP for consistency. For DTDG models, we maintain uniform snapshot intervals across datasets. Remaining parameters are left at default values as they are typically optimal.

Datasets and Metrics: We evaluate on six graph datasets (Wikipedia, Reddit, MOOC, Flights, ML25M, and DGraphFin). Specific details of the datasets can be found in Table 2. In evaluating training accuracy, we consider four key metrics (AUC, AP, Recall, and Accuracy).

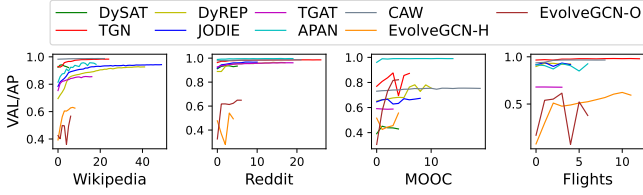


Figure 5: Val.ap after each training epoch.

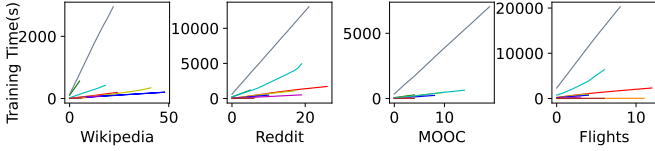


Figure 6: Total training time cost after each epoch.

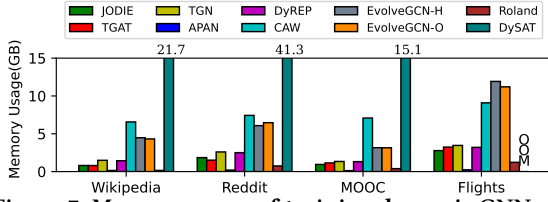


Figure 7: Memory usage of training dynamic GNN models.

Furthermore, we analyze time cost and memory usage to assess training efficiency and scalability. Our primary focus is on the link prediction task, the most common task for dynamic GNNs.

6.2 Comparison of Dynamic GNN Models

We extensively compared the nine dynamic GNN models by training them to convergence. The convergence accuracy are detailed in Table 5. Notably, most models encountered out-of-memory (OOM) issues on the DGraphFin and ML25M datasets, except for APAN and CAW. However, even APAN and CAW exhibited significant slowness and failed to complete within 48 hours. Hence, the results on these two datasets are excluded. Furthermore, we illustrate the convergence curves of validated AP and time cost after each epoch in Figure 5 and 6, respectively. Roland’s distinctive training method allows it to converge after just one epoch, hence its results are not included in these figures. GPU memory usage is shown in Figure 7.

Table 5 highlights the consistent top performance of CTDG models, with TGN, APAN, CAW, and DyREP excelling on various datasets and metrics. CAW stands out on the Wikipedia dataset, showcasing superior performance across all metrics due to its unique anonymous causal walk approach. APAN shines on the Reddit and MOOC datasets, demonstrating versatility beyond financial scenarios into user interaction datasets. TGN excels in AUC/AP on the Flights dataset. We also observed that on the MOOC dataset, EvolveGCN-H has a high Recall value but low Accuracy. This discrepancy is attributed to a high number of false positives (FP), indicating that the model has good discrimination for positive samples, but cannot make correct judgments for negative samples. On the other hand, analysis from Figure 6 and Figure 7 shows that the high training accuracy of CAW and APAN comes with significant time and memory costs, especially for CAW. TGN maintains commendable performance with acceptable training times. DySAT and

EvolveGCN exhibit lower accuracy metrics while consume considerable memory, making them less efficient. Figure 5 indicates that higher validation metrics usually lead to superior test results.

6.3 Comparison of Models on Frameworks

In this section, we evaluate these models trained on dedicated optimized dynamic GNN frameworks with open-sources: TGL, DistTGL, and SPEED. TGL implemented JODIE, TGAT, TGN, APAN, and DySAT models, while DistTGL only focuses on the TGN model. SPEED covers TGN, TGAT, JODIE, DyREP, and TIGE models. For TGL’s DySAT implementation (TGL-DySAT), we maintain the same time interval size as the model for each dataset. We set $top_k = 10$ for SPEED to explore potentially higher evaluation metrics.

Accuracy Metrics Comparison: We show the four accuracy metrics of models trained to converge on frameworks in Figure 8. Situations facing issues such as not-implemented (NI), out-of-memory (OOM), or overtime (OT) are indicated in the corresponding bar. In general, frameworks tend to achieve better accuracy than the origin models, although the model outperforms in some cases. For example, JODIE on the Wikipedia dataset shows superior AUC/AP/Accuracy results compared to the SPEED framework. Frameworks have an advantage over models in their ability to handle large datasets effectively. Models often struggle with training on large datasets, whereas frameworks are tailored to support such scenarios efficiently. Among TGL, SPEED, and DistTGL comparisons, TGL frequently demonstrates higher accuracy.

Training Time Comparison: The training times of these models to reach convergence on various frameworks were compared, with results displayed in Figure 9. It indicates that frameworks generally require less training time compared to the origin models. This trend is especially prominent in TGL, where it consistently outperforms the original on most datasets and models. DistTGL, as a distributed version of TGL, despite having lower evaluation metrics than TGL, shows a reduction in training time. SPEED exhibits a significantly longer training time, with up to 10 times longer than TGL, although this disparity may due to the $top_k = 10$ setting in SPEED.

GPU Memory Usage Comparison: The GPU memory usage during the training of these models on various frameworks was compared, with results displayed in Figure 10. Generally, the origin models tend to consume more memory compared to frameworks, indicating that frameworks are optimized for dynamic GNN training. However, in some cases, SPEED exhibits higher memory usage, possibly due to its graph partitioning approach. Frameworks often utilize specific techniques like TGL/DistTGL’s T-CSR and SPEED’s graph partitioning to accommodate large datasets. Despite this, the memory usage of TGL/DistTGL remains lower than that of SPEED, suggesting the efficiency of TGL/DistTGL’s T-CSR. It’s noteworthy that while ML25M and DGraphFin have similar file sizes (as shown in Table 2), training DGraphFin requires significantly more memory than ML25M due to GNN sensitivity to node-related factors. Additionally, these frameworks do not maximize GPU memory usage, with most cases using only up to 4GB.

6.4 Multi-GPU Scalability

Frameworks generally offer a key advantage over the origin models by incorporating a multi-GPU extension for faster parallel training.

Table 5: Converge accuracy on dynamic GNN models for link prediction task.

Type	Model	Wikipedia				Reddit				MOOC				Flights			
		AUC	AP	Recall	Accuracy	AUC	AP	Recall	Accuracy	AUC	AP	Recall	Accuracy	AUC	AP	Recall	Accuracy
CTDG	JODIE	0.9325	0.9313	0.8657	0.8515	0.9636	0.9610	0.9155	0.9019	0.6243	0.5920	0.6805	0.5942	0.9116	0.9049	0.9395	0.7862
	TGAT	0.8202	0.8339	0.7838	0.7351	0.9585	0.9610	0.9151	0.8877	0.5637	0.5720	0.8419	0.5460	0.7008	0.6800	0.6589	0.6373
	TGN-attn	0.9781	0.9788	0.9076	0.9191	0.9861	0.9862	0.9450	0.9410	0.8288	0.8078	0.6715	0.7403	0.9763	0.9722	0.9833	0.8927
	APAN	0.9650	0.9573	0.9579	0.9094	0.9969	0.9965	0.9886	0.9777	0.9935	0.9883	0.9896	0.9859	0.8995	0.8904	0.9411	0.7818
	CAW	0.9819	0.9854	0.9151	0.9430	0.9835	0.9859	0.9210	0.9393	0.7424	0.7724	0.8799	0.6218	0.9626	0.9663	0.8843	0.9053
	DyRep	0.9171	0.9214	0.8195	0.8293	0.9667	0.9654	0.8934	0.9067	0.6997	0.6505	0.4325	0.6007	0.9214	0.9109	0.9842	0.7094
DDTG	EvolveGCN-H	0.9053	0.6253	0.7899	0.8775	0.7345	0.4938	0.4803	0.8997	0.7255	0.5561	0.9951	0.2906	0.9044	0.5941	0.9033	0.7053
	EvolveGCN-O	0.8820	0.5667	0.7242	0.8832	0.9107	0.6490	0.9101	0.6945	0.9366	0.8214	0.9432	0.7964	0.8775	0.3787	0.8647	0.7272
	Roland	0.8584	0.8748	0.7473	0.8145	0.9329	0.9511	0.7894	0.8454	0.9409	0.9571	0.7981	0.7582	0.8512	0.8599	0.3640	0.5168
	DySAT	0.8668	0.8987	0.9293	0.6513	0.9179	0.9345	0.9471	0.7195	0.4551	0.4336	0.5596	0.4862	OOM			

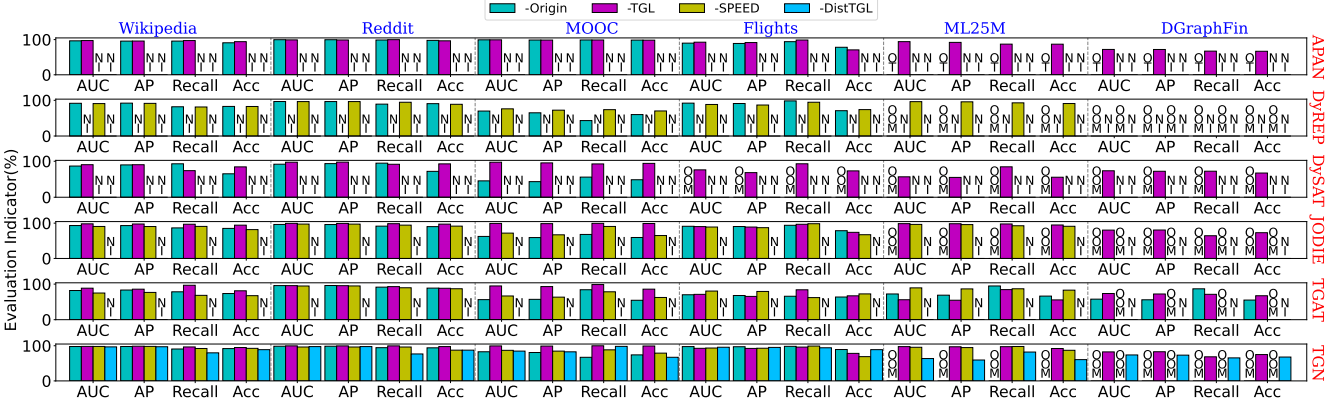


Figure 8: Comparison of accuracy metrics of training models to converge on frameworks. “Acc” means “Accuracy”, “OT” means “OverTime”, “NI” means “Not Implemented”, “OOM” means “Out of Memory”.

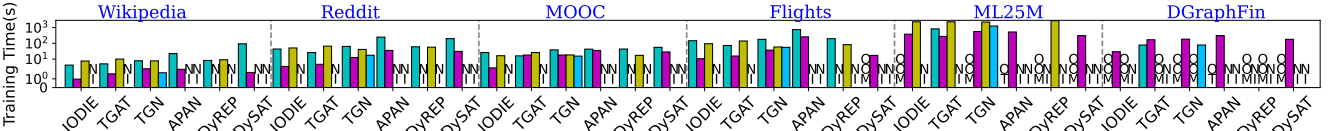


Figure 9: Comparison of average per epoch training time of models and frameworks, and the y-axis are shown in log.

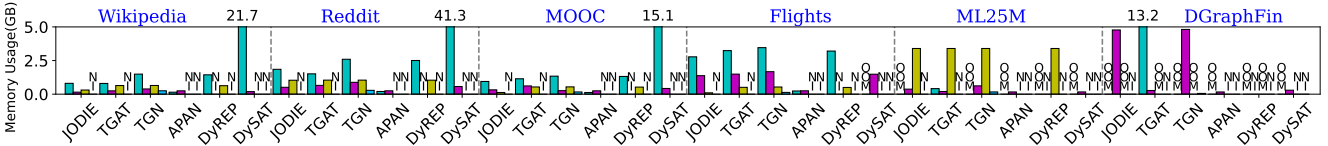


Figure 10: Comparison of GPU memory usage of models and frameworks, with the GPU memory capacity equals 48GB.

In this section, we analyze the performance of different frameworks with varying numbers of GPUs. We assess performance using 1 GPU, 2 GPUs on one machine, 4 GPUs on one machine, and 8 GPUs on two machines. Our analysis is based on the DGraphFin and ML25M datasets, focusing on converged AP, average training time per epoch, and GPU memory usage. Results are presented in Figure 11, Figures 12, and 13, respectively. The lack of data for the 8GPU scenario is due to a lack of multi-machine training support, while the absence for 1GPU is due to out-of-memory issues.

In most scenarios, the AP values demonstrate stability with minimal variation as the number of GPUs increases, suggesting that multi-GPU parallel training has limited impact on model accuracy. An exception is seen with TGL, where a potential decrease in AP occurs as the number of GPUs rises, particularly noticeable in the DGraphFin dataset. This decline is attributed to TGL’s reliance on

mini-batch parallelism, which may overlook dependencies within one-time mini-batches, leading to decreased training accuracy when the total size of one-time mini-batches is doubled. While DistTGL generally shows resilience to changes in GPU numbers, there is a risk of performance deterioration when multiple machines are utilized, as evidenced in the DGraphFin results with 8 GPUs spread across two machines. This suggests that although DistTGL is relatively unaffected by GPU variations, the utilization of multiple machines may introduce accuracy challenges in specific scenarios.

On the other hand, the memory usage for models across the three frameworks tends to rise with an increase in the number of GPUs, while the reduction in average per epoch training time is generally not as pronounced, except for specific instances like transitioning from 1 GPU to 2 GPUs in the ML25M dataset. This trend may be attributed to the impact of multi-GPU communication, which can

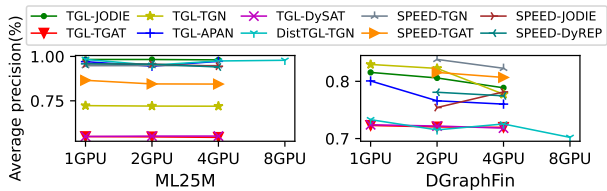


Figure 11: Average precision for different numbers of GPUs.

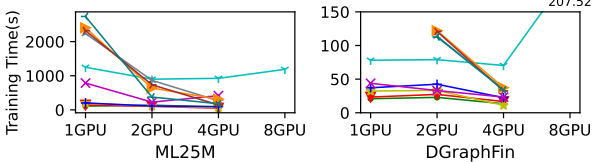


Figure 12: Average epoch time for different numbers of GPUs.

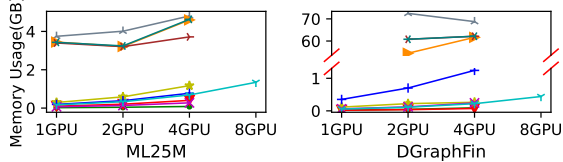


Figure 13: GPU memory usage for different numbers of GPUs. contribute to an overall increase in training time. Interestingly, DistTGL experiences an escalation in average per epoch training time when moving from 4 GPUs (within one machine) to 8 GPUs (across two machines), notably showing a threefold increase in the case of the DGraphFin dataset. This is primarily due to the high cost of communication between multiple machines, resulting in an amplified training time for DistTGL under these configurations.

6.5 Hyperparameter Settings

This section focuses on configuring different batch sizes and GNN layers to assess their impacts on model training performance.

Impact of Batch Size: We experimented with different batch sizes (100, 500, 1000, and 2000) while maintaining a single GNN layer for the six CTDG models. Figure 14 displays the models’ AP results for the different batch sizes, revealing that a batch size of 100 consistently yields better performance. This suggests that, in general, smaller batch sizes can lead to improved indicator results by enabling more efficient feature learning. However, this efficiency may be counterbalanced by longer training times, as depicted in Figure 15. Figure 15 also indicates that an optimal batch size lies between excessively large and excessively small values, representing a compromise. The choice of batch size is typically a trade-off between the number of batches (for smaller sizes) and the computational load per batch (for larger sizes), impacting operational speed. Although TGN, DyRep, and JODIE exhibit decreasing trends up to a batch size of 2000, it is anticipated that training time will escalate beyond a certain batch size threshold. Additionally, as shown in Figure 16, there is a direct correlation between batch size and memory usage, with memory requirements increasing as batch size grows.

Impact of GNN Layers: To investigate the impact of different numbers of GNN layers on model performance, we conducted a two-layer experiment with the TGAT, TGN, APAN, and DyREP models. JODIE was excluded due to the absence of layer concepts, and CAW was prone to out-of-memory (OOM) issues with two layers. The experiments were conducted with a batch size of 1000.

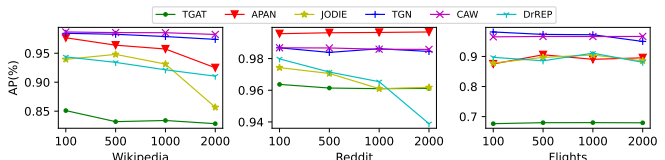


Figure 14: Average precision for different batch sizes.

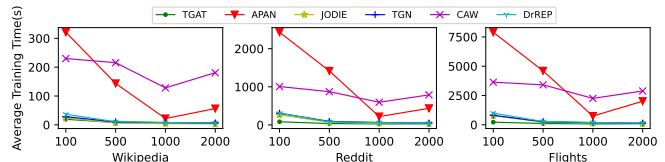


Figure 15: Training time for different batch sizes.

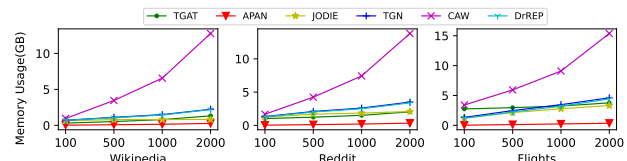


Figure 16: GPU memory usage for different batch sizes.

The evaluation results are presented in Figure 17. The findings suggest that increasing the number of GNN layers can enhance evaluation performance. However, as illustrated in Figures 18 and 19, this improvement comes at the cost of increased training time and memory consumption. The objective is to increase the number of GNN layers to improve performance within a reasonable epoch time while mitigating excessive time and OOM concerns. Notably, APAN appears less sensitive to the number of layers, as its model architecture is not heavily dependent on this factor. Consequently, augmenting the number of layers does not substantially inflate training time or memory usage for APAN. While the results for three layers are not presented, TGAT, TGN, and DyREP encountered OOM issues with three layers, whereas APAN did not. This discrepancy can be attributed to APAN’s model architecture exhibiting minimal correlation with the number of GNN layers.

6.6 Node Classification Evaluation

In this section, we assess the converge accuracy of various dynamic GNN models for the node classification task. Specifically, we consider the JODIE, TGN, DyREP, TGAT, and APAN of CTDG models, as well as the EvolveGCN of the DTDG model. These models are evaluated on the Wikipedia, Reddit, and MOOC datasets, using appropriate labels for node classification. A single GNN layer with a batch size of 100 is utilized for consistency across all models. The results of the AUC and AP metrics are presented in Table 6. Interestingly, the performance metrics of these dynamic GNN models exhibit lower values in node classification tasks, particularly evident in the Reddit and MOOC datasets. This performance difference may stem from these models being primarily designed for the prevalent link prediction task in dynamic graphs. Surprisingly, there is no substantial gap in performance between DTDG and CTDG models in node classification tasks, indicating that both types of dynamic GNN models are similarly effective in this scenario. Notably, TGN attains the highest AUC on the Reddit dataset, while APAN achieves the highest AUC on the Wikipedia dataset. EvolveGCN stands out as the top performer among CTDGs, particularly excelling on the MOOC dataset.

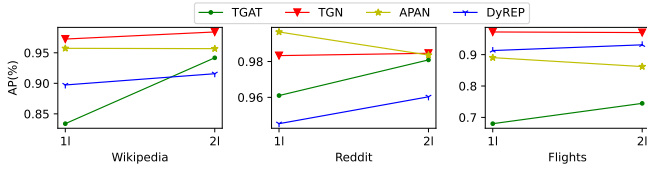


Figure 17: Average precision for different GNN Layers.

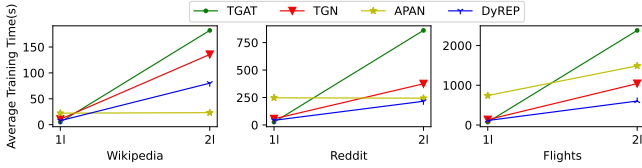


Figure 18: Training time for different GNN Layers.

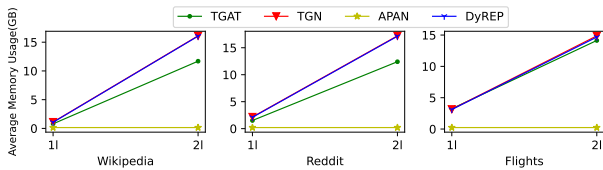


Figure 19: GPU memory usage for different GNN Layers.

Table 6: Converge accuracy for node classification task.

Type	Model	Wikipedia		Reddit		MOOC	
		AUC	AP	AUC	AP	AUC	AP
CTDG	JODIE	0.8021	0.0125	0.6392	0.0033	0.6099	0.0218
	TGAT	0.8571	0.0124	0.6724	0.0019	0.6311	0.0196
	TGN-attn	0.8685	0.0137	0.6821	0.0020	0.5386	0.0162
	APAN	0.8701	0.0141	0.5590	0.0014	0.6412	0.0302
	DyRep	0.8357	0.0130	0.5310	0.0010	0.6561	0.0236
DTDG	EvolveGCN-H	0.6981	0.0148	0.5921	0.0018	0.6792	0.0325
	EvolveGCN-O	0.7920	0.0793	0.5964	0.0021	0.6782	0.0372

7 OPEN CHALLENGES

Despite significant advancements in dynamic GNN models and training frameworks, several open challenges continue to exist, placing them in a phase of ongoing exploration with limitations in diversity, generality, accuracy, efficiency, and scalability.

Diversity in Application Domains. Dynamic graph representation learning is applied in diverse domains such as social networks, transportation networks, epidemic transmission, and recommendation systems, as discussed in §2.1. However, each specific application has unique dynamic graph characteristics, highlighting the need for specialized methods to effectively address the diverse requirements of individual scenarios.

Requirement for a Unified Framework. Existing dynamic graph algorithms utilize various methods to capture time dependencies within graph structures, making it challenging to establish a unified framework. While efforts like TGL [155] have strived to create a unified graph operator, they typically cover only a restricted range of models. Developing a comprehensive unified graph operator capable of encompassing a majority of algorithms is a crucial yet formidable endeavor in the field of dynamic graph learning.

Challenges in Processing Dynamic Graph Updates. While existing dynamic GNN frameworks provide functional support for training modules essential for dynamic graph models, their assistance is constrained, especially in dynamic graph data storage.

Current frameworks commonly utilize structures like CSR or T-CSR, which encounter difficulties in promptly supporting graph updates to real-time dynamic graph updates. This constraint also restricts the adaptability of dynamic graph models.

Challenges in Storing Large-Scale Dynamic Graphs. The data volume of dynamic graph data evolving over time is considerably larger than that of static graphs. Dynamic GNN models also necessitate storage of additional long-term and short-term memory information related to the evolution of graph data, resulting in significant storage and computational requirements. Many existing models are limited in their ability to handle large graphs, and when confronted with such scenarios, they often require substantial GPU resources for processing, indicating scalability challenges. As the size of graphs increases, distributed processing becomes a viable solution [156]. Therefore, there is an urgent need to advance distributed and parallel computing methodologies to effectively manage and process large-scale dynamic graph data.

Inefficient Training Data Extraction. In dynamic GNN training, the extraction of training data involves not only feature data but also historical memory information, which can impede the GNN training efficiency. In distributed parallel training frameworks, data extraction also includes data transmission among multiple GPUs and machines, adding to the inefficiency of the process. Furthermore, updating this memory information post-training further reduces the effectiveness of data extraction in dynamic GNN training. On the other hand, our experiments reveal instances of under-utilization of GPU memory, presenting an opportunity to leverage this available memory space to accelerate the data extraction process.

Challenges in Parallel Training Efficiency. The majority of current frameworks support training on a single machine with single or multiple GPUs but lack distributed environment support across multiple machines. This limitation restricts their scalability in parallel training large dynamic graphs. Additionally, the increasing complexity of dynamic GNN models and temporal dependencies within dynamic graphs pose further obstacles to parallel training efficiency. Current frameworks may either neglect dependencies within individual mini-batches to improve parallel training efficiency, or concentrate solely on capturing temporal dependencies leading to suboptimal training efficiency.

8 CONCLUSION

This paper provides a comprehensive comparative analysis and experimental evaluation of dynamic GNNs. It covers 81 dynamic GNN models with a novel taxonomy, compares 12 dynamic GNN training frameworks, and includes commonly used benchmarks for evaluating dynamic GNNs. The paper includes extensive experiments on nine representative models and three frameworks across six standard graph datasets using unified benchmarks and evaluation metrics. Performance evaluation covers metrics related to convergence accuracy, training efficiency, and GPU memory usage, considering both single GPU and multiple GPUs scenarios. The analysis and evaluation results highlight various open challenges in the dynamic GNN field to offer valuable principles for future researchers to enhance the generality, performance, efficiency, and scalability of dynamic GNN models and frameworks.

REFERENCES

- [1] Anon. 2022. The DBLP datasets. Retrieved April 28, 2024 from <https://www.dropbox.com/sh/palzyh5box1uc1v/AACSLHB7PChT-ruN-rksZTCYa?dl=0>.
- [2] Anon. 2022. The GDELT datasets. Retrieved April 28, 2024 from <https://github.com/amazon-science/tgl/blob/main/down.sh>.
- [3] Anon. unknown. The DGraphFin datasets. Retrieved April 28, 2024 from <https://dgraph.xinye.com/dataset>.
- [4] Anon. unknown. The ML25M datasets. Retrieved April 28, 2024 from <https://grouplens.org/datasets/movielens/25m/>.
- [5] Anon. unknown. The Yelp datasets. Retrieved April 28, 2024 from <https://www.yelp.com/dataset>.
- [6] Guangji Bai, Chen Ling, and Liang Zhao. 2022. Temporal domain generalization with drift-aware dynamic neural networks. *arXiv preprint arXiv:2205.10664* (2022).
- [7] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).
- [8] Anson Bastos, Abhishek Nadgeri, Kuldeep Singh, Toyotaro Suzumura, and Manish Singh. 2023. Learnable spectral wavelets on dynamic graphs to capture global interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 6779–6787.
- [9] Stephen Bonner, Amir Atapour-Abarghouei, Philip T Jackson, John Brennan, Ibad Kureshi, Georgios Theodoropoulos, Andrew Stephen McGough, and Boguslaw Obara. 2019. Temporal neighbourhood aggregation: Predicting future links in temporal graphs via recurrent variational graph convolutions. In *2019 IEEE international conference on big data (Big Data)*. IEEE, 5336–5345.
- [10] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS Coded Event Data. <https://doi.org/10.7910/DVN/28075>
- [11] Guillem Casadesus-Vila, Joan-Adria Ruiz-de Azua, and Eduard Alarcón. 2024. Toward autonomous cooperation in heterogeneous nanosatellite constellations using dynamic graph neural networks. *arXiv preprint arXiv:2403.00692* (2024).
- [12] Venkatesan T Chakaravathy, Shivmaran S Pandian, Saurabh Raje, Yogish Sabharwal, Toyotaro Suzumura, and Shashanka Ubaru. 2021. Efficient scaling of dynamic graph neural networks. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15.
- [13] Chaoyi Chen, Dechao Gao, Yanfeng Zhang, Qiang Wang, Zhenbo Fu, Xuecang Zhang, Junhua Zhu, Yu Gu, and Ge Yu. 2023. NeutronStream: A Dynamic GNN Training Framework with Sliding Window for Graph Streams. *Proceedings of the VLDB Endowment* 17, 3 (2023), 455–468.
- [14] Jinyin Chen, Jian Zhang, Xuanheng Xu, Chenbo Fu, Dan Zhang, Qingpeng Zhang, and Qi Xuan. 2019. E-LSTM-D: A deep learning framework for dynamic network link prediction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 6 (2019), 3699–3712.
- [15] Xi Chen, Yongxiang Liao, Yun Xiong, Yao Zhang, Siwei Zhang, Jiawei Zhang, and Yiheng Sun. 2023. SPEED: Streaming Partition and Parallel Acceleration for Temporal Interaction Graph Embedding. *arXiv preprint arXiv:2308.14129* (2023).
- [16] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
- [17] Andrea Cini, Ivan Marisca, Filippo Maria Bianchi, and Cesare Alippi. 2023. Scalable spatiotemporal graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 7218–7226.
- [18] David R Cox and Peter Adrian Walter Lewis. 1972. Multivariate point processes. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 3. 401–448.
- [19] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2019. Learning dynamic context graphs for predicting social events. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1007–1016.
- [20] Shaohua Fan, Shuyang Zhang, Xiao Wang, and Chuan Shi. 2023. Directed acyclic graph structure learning from dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 7512–7521.
- [21] Lanting Fang, Kaiyu Feng, Jie Gui, Shanshan Feng, and Aiqun Hu. 2023. Anonymous Edge Representation for Inductive Anomaly Detection in Dynamic Bipartite Graph. *Proceedings of the VLDB Endowment* 16, 5 (2023), 1154–1167.
- [22] Yuchen Fang, Yanjun Qin, Haiyong Luo, Fang Zhao, Bingbing Xu, Liang Zeng, and Chenxing Wang. 2023. When spatio-temporal meet wavelets: Disentangled traffic forecasting via efficient spectral graph attention networks. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 517–529.
- [23] Mehrdad Farajtabar, Nan Du, Manuel Gomez Rodriguez, Isabel Valera, Hongyuan Zha, and Le Song. 2014. Shaping social activity by incentivizing users. *Advances in neural information processing systems* 27 (2014).
- [24] Kaituo Feng, Changsheng Li, Xiaolu Zhang, and Jun Zhou. 2023. Towards Open Temporal Graph Neural Networks. *arXiv preprint arXiv:2303.15015* (2023).
- [25] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).
- [26] Kaihua Fu, Quan Chen, Yuzhuo Yang, Jiuchen Shi, Chao Li, and Minyi Guo. 2023. BLAD: Adaptive Load Balanced Scheduling and Operator Overlap Pipeline For Accelerating The Dynamic GNN Training. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–13.
- [27] Swapnil Gandhi and Anand Padmanabha Iyer. 2021. P3: Distributed deep graph learning at scale. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*. 551–568.
- [28] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv:1809.03202* (2018).
- [29] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. 2018. Neural processes. *arXiv preprint arXiv:1807.01622* (2018).
- [30] Wulfram Gerstner and Werner M Kistler. 2002. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.
- [31] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyn-graph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems* 187 (2020), 104816.
- [32] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273* (2018).
- [33] Mingyu Guan, Anand Padmanabha Iyer, and Taesoo Kim. 2022. Dynagraph: dynamic graph neural networks at scale. In *Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*. 1–10.
- [34] Ehsan Hajiramezani, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational graph recurrent neural networks. *Advances in neural information processing systems* 32 (2019).
- [35] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).
- [36] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).
- [37] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2018. The spectral graph wavelet transform: Fundamental theory and fast computation. In *Vertex-Frequency Analysis of Graph Signals*. Springer, 141–175.
- [38] Liangzhe Han, Ruixing Zhang, Leilei Sun, Bowen Du, Yanjie Fu, and Tongyu Zhu. 2023. Generic and dynamic graph representation learning for crowd flow modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 4293–4301.
- [39] Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. 2020. Explainable sub-graph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*.
- [40] Zhen Han, Jindong Jiang, Yuyi Wang, Yunpu Ma, and Volker Tresp. 2019. The graph hawks network for reasoning on temporal knowledge graphs. In *Learning with Temporal Point Processes Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019) NeurIPS 2019*.
- [41] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.
- [42] Bowei He, Xu He, Yingxue Zhang, Ruiming Tang, and Chen Ma. 2023. Dynamically Expandable Graph Convolution for Streaming Recommendation. *arXiv preprint arXiv:2303.11700* (2023).
- [43] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [44] Junfeng Hu, Yuxuan Liang, Zhencheng Fan, Hongyang Chen, Yu Zheng, and Roger Zimmermann. 2023. Graph Neural Processes for Spatio-Temporal Extrapolation. *arXiv preprint arXiv:2305.18719* (2023).
- [45] Wenjie Hu, Yang Yang, Ziqiang Cheng, Carl Yang, and Xiang Ren. 2021. Time-series event prediction with evolutionary state graph. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 580–588.
- [46] Xuanwen Huang, Yang Yang, Yang Wang, Chungung Wang, Zhisheng Zhang, Jiarong Xu, Lei Chen, and Michalis Vazirgiannis. 2022. Dgraph: A large-scale financial dataset for graph anomaly detection. *Advances in Neural Information Processing Systems* 35 (2022), 22765–22777.
- [47] Herbert Jaeger. 2007. Echo state network. *scholarpedia* 2, 9 (2007), 2330.
- [48] Shuo Ji, Xiaodong Lu, Mingzhe Liu, Leilei Sun, Chuanren Liu, Bowen Du, and Hui Xiong. 2023. Community-based Dynamic Graph Learning for Popularity Prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 930–940.
- [49] Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. 2020. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems* 2 (2020), 187–198.
- [50] Renhe Jiang, Zhaonan Wang, Jiawei Yong, Puneet Jeph, Quanjun Chen, Yasumasa Kobayashi, Xuan Song, Shintaro Fukushima, and Toyotaro Suzumura. 2023. Spatio-temporal meta-graph learning for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 8078–8086.

- [51] Guangyin Jin, Yuxuan Liang, Yuchen Fang, Zezhi Shao, Jincui Huang, Junbo Zhang, and Yu Zheng. 2023. Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2023).
- [52] Guangyin Jin, Lingbo Liu, Fuxian Li, and Jincui Huang. 2023. Spatio-Temporal Graph Neural Point Process for Traffic Congestion Event Prediction. (2023).
- [53] W Jin, C Zhang, PA Szekeley, and X Ren. [n.d.]. Recurrent Event Network for Reasoning over Temporal Knowledge Graphs. arXiv 2019. *arXiv preprint arXiv:1904.05530* ([n. d.]).
- [54] Mikel Joaristi and Edoardo Serra. 2021. Sir-gn: A fast structural iterative representation learning approach for graph nodes. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 6 (2021), 1–39.
- [55] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupard. 2020. Representation learning for dynamic graphs: A survey. *The Journal of Machine Learning Research* 21, 1 (2020), 2648–2720.
- [56] Samira Khodabandehlou and Alireza Hashemi Golpayegani. 2024. FiFraud: Unsupervised Financial Fraud Detection in Dynamic Graph Streams. *ACM Transactions on Knowledge Discovery from Data* 18, 5 (2024), 1–29.
- [57] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [58] Bryan Klimt and Yiming Yang. 2004. Introducing the Enron corpus.. In *CEAS*, Vol. 45. 92–96.
- [59] Boris Knyazev, Carolyn Augusta, and Graham W Taylor. 2021. Learning temporal attention in dynamic graphs with bilinear interactions. *Plos one* 16, 3 (2021), e0247936.
- [60] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. 2018. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 333–341.
- [61] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, IEEE, 221–230.
- [62] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1269–1278.
- [63] Jérôme KUNEGIS. 2017. The UCI datasets. Retrieved April 28, 2024 from <http://konect.cc/networks/opsahl-ucforum/>.
- [64] Janet Layne, Justin Carpenter, Edoardo Serra, and Francesco Gullo. 2023. Temporal sir-gn: Efficient and effective structural representation learning for temporal graphs. *Proceedings of the VLDB Endowment* 16, 9 (2023), 2075–2089.
- [65] Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*, Vol. 2. Citeseer, 1–49.
- [66] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2005. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 177–187.
- [67] Hongxi Li, Zuxuan Zhang, Dengzhe Liang, and Yuncheng Jiang. 2024. K-Truss Based Temporal Graph Convolutional Network for Dynamic Graphs. In *Asian Conference on Machine Learning*. PMLR, 739–754.
- [68] Jia Li, Zhichao Han, Hong Cheng, Jiao Su, Pengyun Wang, Jianfeng Zhang, and Lujia Pan. 2019. Predicting path failure in time-evolving graphs. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1279–1289.
- [69] Jintang Li, Zhouxin Yu, Zulun Zhu, Liang Chen, Qi Yu, Zibin Zheng, Sheng Tian, Ruofan Wu, and Changhua Meng. 2023. Scaling up dynamic graph representation learning via spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 8588–8596.
- [70] Xiaohan Li, Mengqi Zhang, Shu Wu, Zheng Liu, Liang Wang, and S Yu Philip. 2020. Dynamic graph collaborative filtering. In *2020 IEEE international conference on data mining (ICDM)*. IEEE, 322–331.
- [71] Yiming Li, Yanyan Shen, Lei Chen, and Mingxuan Yuan. 2023. Zebra: When temporal graph neural networks meet temporal personalized pagerank. *Proceedings of the VLDB Endowment* 16, 6 (2023), 1332–1345.
- [72] Ke Liang, Lingyuan Meng, Meng Liu, Yue Liu, Wenxuan Tu, Siwei Wang, Sihang Zhou, and Xinwang Liu. 2023. Learn from relational correlations and periodic events for temporal knowledge graph reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1559–1568.
- [73] Jie Liu, Xuequn Shang, Xiaolin Han, Wentao Zhang, and Hongzhi Yin. 2024. Spatial-temporal Memories Enhanced Graph Autoencoder for Anomaly Detection in Dynamic Graphs. *arXiv preprint arXiv:2403.09039* (2024).
- [74] Kangzheng Liu, Feng Zhao, Guangdong Xu, Xianzhi Wang, and Hai Jin. 2023. RETIA: relation-entity twin-interact aggregation for temporal knowledge graph extrapolation. In *IEEE International Conference on Data Engineering*. IEEE.
- [75] Ye Liu, Hui Li, Alberto Garcia-Duran, Mathias Niepert, Daniel Onoro-Rubio, and David S Rosenblum. 2018. The ICEWS14/05-15 datasets. Retrieved April 28, 2024 from <https://github.com/mniepert/mmkb/tree/master/TemporalKGs>.
- [76] Antonio Longa, Veronica Lachi, Gabriele Santin, Monica Bianchini, Bruno Lepri, Pietro Lio, Franco Scarselli, and Andrea Passerini. 2023. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *arXiv preprint arXiv:2302.01018* (2023).
- [77] Yuanfu Lu, Xiao Wang, Chuan Shi, Philip S Yu, and Yanfang Ye. 2019. Temporal network embedding with micro-and macro-dynamics. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 469–478.
- [78] Yuhong Luo and Pan Li. 2022. Neighborhood-aware scalable temporal network representation learning. In *Learning on Graphs Conference*. PMLR, 1–1.
- [79] Lingxiao Ma, Zhi Yang, Youshan Miao, Jilong Xue, Ming Wu, Lidong Zhou, and Yafei Dai. 2019. {NeuGraph}: Parallel deep neural network computation on large graphs. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. 443–458.
- [80] Yao Ma, Ziyi Guo, Zhaocun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming graph neural networks. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 719–728.
- [81] Anmol Madan, Manuel Cebrian, Sai Moturu, Katayoun Farrahi, et al. 2008. The Social Evolution datasets. Retrieved April 28, 2024 from <http://realitycommons.media.mit.edu/socialevolution.html>.
- [82] Anmol Madan, Manuel Cebrian, Sai Moturu, Katayoun Farrahi, et al. 2011. Sensing the "health state" of a community. *IEEE Pervasive Computing* 11, 4 (2011), 36–45.
- [83] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognition* 97 (2020), 107000.
- [84] Giang H Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunye Koh, and Sungchul Kim. 2018. Dynamic network embeddings: From random walks to temporal random walks. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 1085–1092.
- [85] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. 2009. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 911–932.
- [86] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 601–610.
- [87] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Scharld, and Charles Leiserson. 2020. Evolvegn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 5363–5370.
- [88] Ana Pavlicic. 2009. The Bitcoin-Alpah/Bitcoin-OTC/AS-773 datasets. Retrieved April 28, 2024 from <https://snap.stanford.edu/data>.
- [89] Ana Pavlicic. 2009. The Wikipedia/Reddit/MOOC/LastFM datasets. Retrieved April 28, 2024 from <http://snap.stanford.edu/jodie>.
- [90] Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, , and Reihaneh Rabbany. 2022. The Enron/Flights datasets. Retrieved April 28, 2024 from <https://zenodo.org/records/7213796#.Y1cO6y8r30o>.
- [91] Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. 2022. Towards better evaluation for dynamic link prediction. *Advances in Neural Information Processing Systems* 35 (2022), 32928–32941.
- [92] Xiao Qin, Nasrullah Sheikh, Chuan Lei, Berthold Reinwald, and Giacomo Domeniconi. 2023. SEIGN: A Simple and Efficient Graph Neural Network for Large Dynamic Graphs. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2850–2863.
- [93] Liang Qu, Huaisheng Zhu, Qiqi Duan, and Yuhui Shi. 2020. Continuous-time link prediction via temporal dependent graph neural network. In *Proceedings of The Web Conference 2020*. 3026–3032.
- [94] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. *Advances in neural information processing systems* 20 (2007).
- [95] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).
- [96] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.
- [97] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Facebook datasets. Retrieved April 28, 2024 from <https://networkrepository.com/fb-wosm-friends.php>.
- [98] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The FB-FORUM datasets. Retrieved April 28, 2024 from <https://networkrepository.com/fb-forum.php>.
- [99] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, et al. 2021. Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 4564–4573.
- [100] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. 2021. A Gentle Introduction to Graph Neural Networks. *Distill* (2021). <https://doi.org/10.23915/distill.00033> <https://distill.pub/2021/gnn-intro>.

- [101] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th international conference on web search and data mining*. 519–527.
- [102] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer, 593–607.
- [103] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part I 25*. Springer, 362–373.
- [104] Sakr Shreif, Bonifati Angela, Voigt Hannes, and Iosup Alexandru. 2021. The Future is Big Graphs: A Community View on Graph Processing Systems. *Communications of ACM* 64, 9 (2021).
- [105] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. 2021. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access* 9 (2021), 79143–79168.
- [106] Junwei Su, Difan Zou, and Chuan Wu. 2024. PRES: Toward Scalable Memory-Based Dynamic Graph Neural Networks. *arXiv preprint arXiv:2402.04284* (2024).
- [107] Susheel Suresh, Mayank Shrivastava, Arko Mukherjee, Jennifer Neville, and Pan Li. 2023. Expressive and Efficient Representation Learning for Ranking Links in Temporal Graphs. In *Proceedings of the ACM Web Conference 2023*. 567–577.
- [108] Haoran Tang, Shiqing Wu, Guandong Xu, and Qing Li. 2023. Dynamic graph evolution learning for recommendation. In *Proceedings of the 46th international acm sigir conference on research and development in information retrieval*. 1589–1598.
- [109] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 990–998.
- [110] Yuxing Tian, Yiyan Qi, and Fan Guo. 2023. FreeDyG: Frequency Enhanced Continuous-Time Dynamic Graph Model for Link Prediction. In *The Twelfth International Conference on Learning Representations*.
- [111] Domenico Tortorella and Alessio Micheli. 2021. Dynamic graph echo state networks. *arXiv preprint arXiv:2110.08565* (2021).
- [112] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *international conference on machine learning*. PMLR, 3462–3471.
- [113] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*.
- [114] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [115] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [116] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P Gummadi. 2009. On the evolution of user interaction in facebook. In *Proceedings of the 2nd ACM workshop on Online social networks*. 37–42.
- [117] Cheng Wan, Youjie Li, Cameron R Wolfe, Anastasios Kyrillidis, Nam Sung Kim, and Yingyan Lin. 2022. PipeGCN: Efficient full-graph training of graph convolutional networks with pipelined feature communication. *arXiv preprint arXiv:2203.10428* (2022).
- [118] Chunyang Wang, Desen Sun, and Yuebin Bai. 2023. PiPAD: pipelined and parallel dynamic GNN training on GPUs. In *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*. 405–418.
- [119] Junshan Wang, Wenhao Zhu, Guojie Song, and Liang Wang. 2022. Streaming graph neural networks with generative replay. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1878–1888.
- [120] Minjie Yu Wang. 2019. Deep graph library: Towards efficient and scalable deep learning on graphs. In *ICLR workshop on representation learning on graphs and manifolds*.
- [121] Xuhong Wang, Ding Lyu, Mengjian Li, Yang Xia, Qi Yang, Xinwen Wang, Xinguang Wang, Ping Cui, Yupu Yang, Bowen Sun, et al. 2021. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *Proceedings of the 2021 international conference on management of data*. 2628–2638.
- [122] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive representation learning in temporal networks via causal anonymous walks. *arXiv preprint arXiv:2101.05974* (2021).
- [123] Yanbang Wang, Pan Li, Chongyang Bai, and Jure Leskovec. 2021. TEDIC: Neural modeling of behavioral patterns in dynamic social interaction networks. In *Proceedings of the Web Conference 2021*. 693–705.
- [124] Yanbang Wang, Pan Li, Chongyang Bai, V Subrahmanian, and Jure Leskovec. 2020. Generic representation learning for dynamic social interaction. In *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining Workshop*.
- [125] Cheng Wu, Chaokun Wang, Jingcao Xu, Ziwei Fang, Tiankai Gu, Changping Wang, Yang Song, Kai Zheng, Xiaowei Wang, and Guorui Zhou. 2023. Instant Representation Learning for Recommendation over Large Dynamic Graphs. *arXiv preprint arXiv:2305.18622* (2023).
- [126] Jiapeng Wu, Meng Cao, Jackie Chi Kit Cheung, and William L Hamilton. 2020. Temp: Temporal message passing for temporal knowledge graph completion. *arXiv preprint arXiv:2010.03526* (2020).
- [127] Yuxia Wu, Yuan Fang, and Lizi Liao. 2024. On the Feasibility of Simple Transformer for Dynamic Graph Modeling. *arXiv preprint arXiv:2401.14009* (2024).
- [128] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.
- [129] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kamran Achan. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).
- [130] Hansheng Xue, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Yu Lin. 2021. Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*. Springer, 282–298.
- [131] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. 2021. Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1975–1985.
- [132] Mingxia Yang, Han Zhu, Tingting Wang, Jijing Cai, Xiang Weng, Hailin Feng, and Kai Fang. 2024. Vehicle Interactive Dynamic Graph Neural Network Based Trajectory Prediction for Internet of Vehicles. *IEEE Internet of Things Journal* (2024).
- [133] Qiang Yang, Changsheng Ma, Qiannan Zhang, Xin Gao, Chuxu Zhang, and Xiangliang Zhang. 2023. Interpretable Research Interest Shift Detection with Temporal Heterogeneous Graphs. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 321–329.
- [134] Shuang-Hong Yang and Hongyuan Zha. 2013. Mixture of mutually exciting processes for viral diffusion. In *International Conference on Machine Learning*. PMLR, 1–9.
- [135] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2358–2366.
- [136] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenberg, and Jure Leskovec. 2019. Hierarchical temporal convolutional networks for dynamic recommender systems. In *The world wide web conference*. 2236–2246.
- [137] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).
- [138] Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. 2024. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems* 36 (2024).
- [139] Haonan Yuan, Qingyun Sun, Xingcheng Fu, Ziwei Zhang, Cheng Ji, Hao Peng, and Jianxin Li. 2024. Environment-Aware Dynamic Graph Learning for Out-of-Distribution Generalization. *Advances in Neural Information Processing Systems* 36 (2024).
- [140] Chuxu Zhang, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V Chawla. 2019. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 1409–1416.
- [141] Haozhen Zhang, Le Yu, Xi Xiao, Qing Li, Francesco Mercaldo, Xiapu Luo, and Qixu Liu. 2023. TFE-GNN: A Temporal Fusion Encoder Using Graph Neural Networks for Fine-grained Encrypted Traffic Classification. In *Proceedings of the ACM Web Conference 2023*. 2066–2075.
- [142] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018).
- [143] Peiyan Zhang, Yuchen Yan, Chaozhao Li, Senzhang Wang, Xing Xie, Guojie Song, and Sunghun Kim. 2023. Continual Learning on Dynamic Graphs via Parameter Isolation. *arXiv preprint arXiv:2305.13825* (2023).
- [144] Xin Zhang, Yanyan Shen, Yingxia Shao, and Lei Chen. 2023. DUCATI: A Dual-Cache Training System for Graph Neural Networks on Giant Graphs with the GPU. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–24.
- [145] Yao Zhang, Yun Xiong, Yongxiang Liao, Yiheng Sun, Yucheng Jin, Xuehao Zheng, and Yangyong Zhu. 2023. TIGER: Temporal Interaction Graph Embedding with Restarts. *arXiv preprint arXiv:2302.06057* (2023).
- [146] Zeyang Zhang, Xin Wang, Ziwei Zhang, Zhou Qin, Weigao Wen, Hui Xue, Haoyang Li, and Wenwu Zhu. 2024. Spectral invariant learning for dynamic graphs under distribution shifts. *Advances in Neural Information Processing Systems* 36 (2024).
- [147] Zeyang Zhang, Ziwei Zhang, Xin Wang, Yijian Qin, Zhou Qin, and Wenwu Zhu. 2023. Dynamic heterogeneous graph attention neural architecture search.

- In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11307–11315.
- [148] Kesen Zhao and Liang Zhang. 2023. Causality-Inspired Spatial-Temporal Explanations for Dynamic Graph Neural Networks. In *The Twelfth International Conference on Learning Representations*.
- [149] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems* 21, 9 (2019), 3848–3858.
- [150] Ziwei Zhao, Xi Zhu, Tong Xu, Aakas Lizhiyu, Yu Yu, Xueying Li, Zikai Yin, and Enhong Chen. 2023. Time-interval Aware Share Recommendation via Bi-directional Continuous Time Dynamic Graphs. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 822–831.
- [151] Shangfei Zheng, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Wei Chen, and Lei Zhao. 2023. DREAM: Adaptive Reinforcement Learning based on Attention Mechanism for Temporal Knowledge Graph Reasoning. *arXiv preprint arXiv:2304.03984* (2023).
- [152] Zetao Zheng, Jie Shao, Jia Zhu, and Heng Tao Shen. 2023. Relational Temporal Graph Convolutional Networks for Ranking-Based Stock Prediction. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 123–136.
- [153] Yuchen Zhong, Guangming Sheng, Tianzuo Qin, Minjie Wang, Quan Gan, and Chuan Wu. 2023. GNNFlow: A Distributed Framework for Continuous Temporal GNN Learning on Dynamic Graphs. *arXiv preprint arXiv:2311.17410* (2023).
- [154] Fan Zhou, Xovee Xu, Ce Li, Goce Trajcevski, Ting Zhong, and Kunpeng Zhang. 2020. A heterogeneous dynamical graph neural networks approach to quantify scientific impact. *arXiv preprint arXiv:2003.12042* (2020).
- [155] Hongkuan Zhou, Da Zheng, Israt Nisa, Vasileios Ioannidis, Xiang Song, and George Karypis. 2022. Tgl: A general framework for temporal gnn training on billion-scale graphs. *arXiv preprint arXiv:2203.14883* (2022).
- [156] Hongkuan Zhou, Da Zheng, Xiang Song, George Karypis, and Viktor Prasanna. 2023. DistTGL: Distributed Memory-Based Temporal Graph Neural Network Training. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12.
- [157] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.
- [158] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. Aligraph: A comprehensive graph neural network platform. *arXiv preprint arXiv:1902.08730* (2019).
- [159] Yifan Zhu, Fangpeng Cong, Dan Zhang, Wenwen Gong, Qika Lin, Wenzheng Feng, Yuxiao Dong, and Jie Tang. 2023. WinGNN: Dynamic Graph Neural Networks with Random Gradient Aggregation Window. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3650–3662.
- [160] Yuecai Zhu, Fuyuan Lyu, Chengming Hu, Xi Chen, and Xue Liu. 2022. Encoder-Decoder Architecture for Supervised Dynamic Graph Learning: A Survey. *arXiv preprint arXiv:2203.10480* (2022).