# Natural and Artificial Dynamics in Graphs: Concept, Progress, and Future

**Dongqi Fu and Jingrui He**

*University of Illinois at Urbana-Champaign, Champaign Urbana, Illinois, USA*

Correspondence*:
Jingrui He
jingrui@illinois.edu

## ABSTRACT

Graph structures have attracted much research attention for carrying complex relational information. Based on graphs, many algorithms and tools are proposed and developed for dealing with real-world tasks such as recommendation, fraud detection, molecule design, etc. In this paper, we first discuss three topics of graph research, i.e., graph mining, graph representations, and graph neural networks (GNNs). Then, we introduce the definitions of *natural dynamics* and *artificial dynamics* in graphs, and the related works of natural and artificial dynamics about how they boost the aforementioned graph research topics, where we also discuss the current limitation and future opportunities.

**Keywords: Graph Mining, Graph Representations, Graph Neural Networks, Natural Dynamics, Artificial Dynamics**

## 1 INTRODUCTION

In the era of big data, the relationship between entities becomes much more complex than ever before. As a kind of relational data structure, graph (or network) attract much research attention for dealing with this unprecedented phenomenon. To be specific, many graph-based algorithms and tools are proposed, such as DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), node2vec (Grover and Leskovec, 2016), GCN (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Velickovic et al., 2018), etc. Correspondingly, many challenges of real-world applications get addressed to some extent, such as recommendation (Fan et al., 2019), fraud detection (Wang et al., 2019), and molecule design (Liu et al., 2018), to name a few.

To investigate graph-based research and relevant problems and applications systematically, at least [1] three aspects will be discussed, i.e., graph mining, graph representations, and graph neural networks (GNNs). Their dependency is convoluted, the reason why we aim to disentangle it is that we can discuss the current efforts from natural and artificial dynamics studies (which are improving the graph algorithms and tools performance) in a fine-grained view, such that we can envision detailed future research opportunities. As for **natural dynamics in graphs**, we use this term to illustrate that the input graphs themselves are evolving, i.e., the topology structures, the node-level, edge-level, and (sub)graph-level features and labels are dependent on time (Aggarwal and Subbian, 2014; Kazemi et al., 2020). As for **artificial dynamics in graphs**, we use this term to describe that end-users *change (e.g., filter, mask, drop, or augment) the existing* or *construct (i.e., from scratch) the non-existing* graph-related elements (e.g., graph topology, graph stream,

[1] Research topics like graph theory and graph database management are also very important, but we skip discussing them in this paper.

node/graph attributes/labels, GNN gradients, GNN layer connections, etc.) to realize the certain performance upgrade (e.g., computation efficiency (Fu et al., 2020b), model explanation (Fu and He, 2021b), decision accuracy (Zheng et al., 2022), etc.). To the best of our knowledge, the first relevant act of conceiving artificial dynamics in graphs appeared in (Kamvar et al., 2003), where "artificial jump" is proposed to adjust the graph topology for PageRank realizing the personal ranking function on structured data, i.e., a random surfer would follow an originally non-existing but newly-added highway to jump to a personally-selected node with a predefined teleportation probability.

With the above introduction of graph research terminology and dynamics category, in this paper, we are ready to introduce some related works on investigating natural and artificial dynamics in graph mining, graph representations, and graph neural networks, and then discuss future research opportunities. To be specific, this survey is organized as follows. The definition and relation introduction for graph mining tasks, graph representations, and graph neural networks are discussed in Section 2. Then, in Section 3, we discuss the formal definition followed by concrete research works for *natural dynamics*, *artificial dynamics*, and *natural + artificial dynamics* in graphs. Finally, in Section 4, we conclude the paper with sharing some research future directions.

## 2 RELATIONS AMONG GRAPH MINING, GRAPH REPRESENTATIONS, AND GRAPH NEURAL NETWORKS

To pave the way for investigating the natural and artificial dynamics in graphs, we first introduce graph research topics (i.e., graph mining, graph representations, and graph neural networks) and their relationships in this section. Then, in the next section, we can target each topic and see how natural dynamics and artificial dynamics contribute to them.

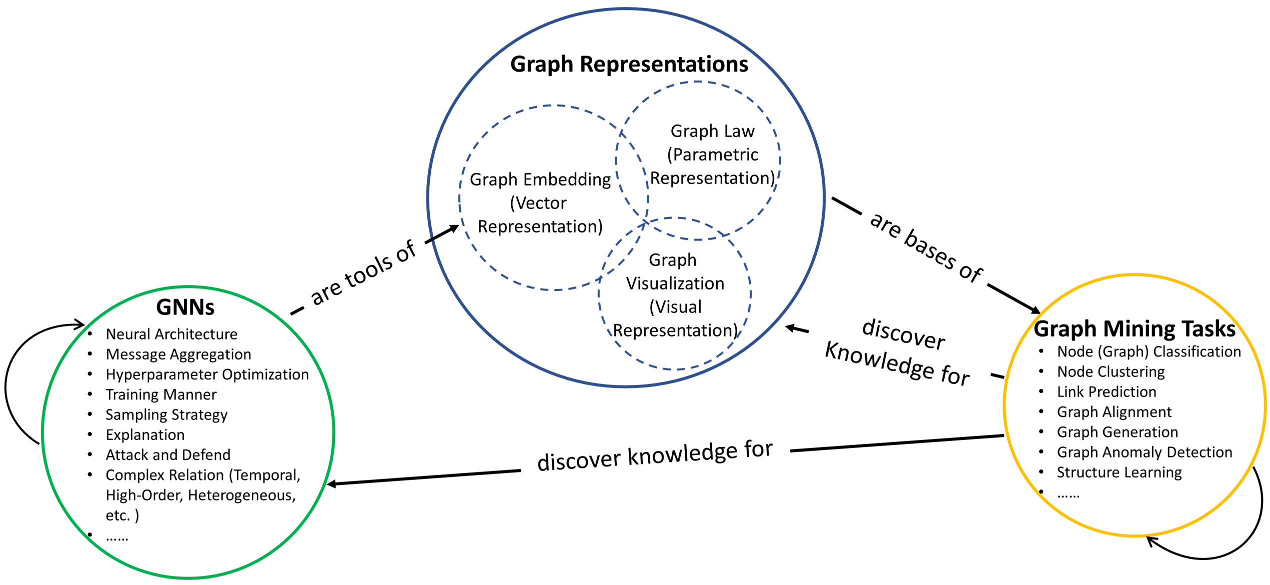


**Figure 1.** Relationships among Graph Mining, Graph Representations, and Graph Neural Networks.

In general, the relationships between graph mining, graph representations, and graph neural networks can be illustrated as shown in Figure 1. (1) Graph mining aims to extract interesting (e.g., non-trivial, implicit, previously unknown, and potentially useful) knowledge from graph data. Graph mining consists of numerous specific tasks, such like node classification (Kipf and Welling, 2017) is aiming to classify the node category based on its features, and node clustering (Shi and Malik, 2000; Andersen et al., 2006) is aiming to partition the entire graph into disjoint or overlapped clusters (i.e., subgraphs) based on end-users' objectives (e.g., conductance, betweeness, etc.):. For example, clustering can discover knowledge to help GNN implementations, and Cluster-GCN (Chiang et al., 2019) is proposed to sample nodes in a topology-preserved clustering, which could entitle vanilla GCN (Kipf and Welling, 2017) the fast computation to deal with large-scale graph datasets. (2) Graph representations are the bases of graph mining, which projects graphs into a proper space such that graph mining can do various task-specific computations. To the best of our knowledge, graph representations consists of three components. First, graph embedding represents graphs with affinity matrices like Laplacian matrix and hidden feature representation matrix, on which many mining tasks rely, such as node classification (Kipf and Welling, 2017); Second, graph law represents graphs with several parameters which describe the statistical property of graphs such as node degree distribution and edge connection probability, which could help mining tasks like graph generation (Leskovec and Faloutsos, 2007) and link prediction (Wang et al., 2021b); Third, graph visualization provides the visual representations and can serve for the domain-specific knowledge interpretation (Bach et al., 2015; Yang et al., 2020a). Within graph representations, graph embedding, graph law, and graph visualization can contribute to each other, and detailed overlapping works are discussed in the following sections. (3) Graph Neural Network (GNN) is an effective tool for extracting meaningful graph embedding vectors (or matrices) by combining deep learning theory and graph theory (Wu et al., 2021). GNNs are composed of a family of many specific models with different research concerns like neural architecture (Chen et al., 2020b) and message passing aggregation design (Klicpera et al., 2019), the detailed related works are also discussed in the following sections.

## 2.1 Graph Mining

Graph mining interacts with real-world problems by discovering knowledge for many applications. Based on structured data, graph mining consists of numerous specific tasks. For example,

- Node (and Graph) Classification (Kipf and Welling, 2017; Zhang et al., 2018; Jing et al., 2021): Nodes sharing similar features should be classified into the same category.
- Node Clustering (or Graph Partitioning) (Spielman and Teng, 2013; Andersen et al., 2006; Shi and Malik, 2000; Ng et al., 2001): Individual nodes are clustered for optimizing certain metrics such as inter-cluster distance, intra-cluster density, etc.
- Link Prediction (Dunlavy et al., 2011; Zhang and Chen, 2018; Kumar et al., 2019): The probability is estimated that whether two nodes should be connected based on evidence like node structural and attribute similarity.
- Graph Generation (Leskovec and Faloutsos, 2007; You et al., 2018; Bojchevski et al., 2018; Zhou et al., 2019, 2020): Model the distribution of a batch of observed graphs and then generate new graphs.
- Subgraph Matching (Tong et al., 2007; Zhang et al., 2009; Du et al., 2017; Liu et al., 2021): Check whether a query graph (usually the smaller one) can be matched in a data graph (usually the larger one) approximately or exactly.
- Graph Anomaly Detection (Akoglu et al., 2015; Yu et al., 2018b; Zheng et al., 2019): Identify whether the graph has abnormal entities like nodes, edges, subgraphs, etc.
- Graph Alignment (Zhang and Tong, 2016; Zhou et al., 2021; Yan et al., 2021b,a): Retrieve similar structures (e.g., nodes, edges, and subgraphs) across graphs.
- many more ...

Those tasks can be directly adapted to solve many high-impact problems in real-world settings. For example, through learning the graph distribution and adding specific domain knowledge constraints, graph generators could contribute to molecule generation and drug discovery (Luo and Ji, 2022; Liu et al., 2022a); With modeling picture pixels as nodes, graph partitioning algorithms could achieve effective image segmentation at scale (Bianchi et al., 2020); By modeling the information dissemination graph over news articles, readers, and publishers (Nguyen et al., 2020) or modeling the suspicious articles into word graphs (Fu et al., 2022a), node and graph classification tasks can help detect fake news in the real world.

## 2.2 Graph Representations

For accomplishing various graph mining tasks, graph representations are indispensable for providing the bases for task-specific computations. To the best of our knowledge, graph representations can be roughly categorized into three aspects, (1) graph embedding (i.e., vector representation), (2) graph law (i.e., parametric representation), and (3) graph visualization (i.e., visual representation).

### 2.2.1 Graph Embedding (Vector Representation)

**First, graph representations can be in the form of embedding matrices**, i.e., the graph topological information and attributes are encoded into a matrix (or matrices). The most common form can be the Laplacian matrix, which is the combination of the graph adjacency matrix and degree matrix. Recently, the graph embedding (or graph representation learning) area attracts many research interests, along with numerous graph embedding methods proposed for extracting the node (or graph) hidden representation vectors from the input affinity matrices, like DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), and node2vec (Grover and Leskovec, 2016). They [2] share the general principle to extract node representation vectors, which means a node could reflect (e.g., predict, be proximate to, etc.) its sampled neighbors in the embedding space, e.g., Skip-gram in (Perozzi et al., 2014; Grover and Leskovec, 2016) and order-based proximity in (Tang et al., 2015). With the different angles of viewing graph topology and node features, some derivatives are proposed, such as metapath2vec (Dong et al., 2017) for heterogeneous networks, graph2vec (Narayanan et al., 2017) for the graph-level embeddings, and tdGraphEmbd (Beladev et al., 2020) for temporal graph-level embeddings.

All graph embedding works mentioned above are unsupervised, which means the guidance (or constraints, regularizers) during the learning process are totally from the input graph structure and features, such that the encoded vectors within specific dimensions are actually reflecting the graph itself information. Hence, by involving extra domain knowledge (i.e., labels and task-specific loss functions), graph embedding vectors can serve real-world applications. For example, with user-item interaction history records and user anomaly labels, graph embedding techniques can be leveraged for predicting the user-interested merchandise and user's behavior in the future (Kumar et al., 2019); By involving additional labels, graph embedding vectors can be used to generate small molecule graphs through an encoder-decoder framework (Simonovsky and Komodakis, 2018; Jin et al., 2018); Also, with delicately designed query questions and temporal knowledge graphs, graph embedding techniques can be used to help answer open-world questions (Saxena et al., 2021; Shang et al., 2022).

[2] As another kind of powerful tool for graph embedding, graph neural networks (GNNs) become popular and attract research attention from both the deep learning domain and graph theory domain. Here, "they" are not referring to graph neural networks. And we set up another section for introducing GNNs, otherwise Section 2.2 will be enormous and overstaffing. GNNs will be discussed separately in Section 2.3. We would like to note that GNN is a tool for realizing graph embedding as we illustrated in Figure 1, the context separation in the paper is not standing for the tied hierarchy of graph representations and graph neural networks.

#### 2.2.2 Graph Law (Parametric Representation)

**Second, graphs can also be represented by several parameters**. A simple but common example is Erdős-Rényi random graph, i.e., $G(n, p)$ or $G(n, m)$ (Drobyshevskiy and Turdakov, 2020). To be specific, in $G(n, p)$, the possibility of establishing a single edge among $n$ nodes is independent of each other and valued by a constant parameter $p$; while in $G(n, m)$, an $n$-node and $m$-edge graph is chosen evenly from all possible $n$-node and $m$-edge graph collections. In addition to the number of nodes, the number of edges, and the edge probability, many common parameters are well-studied for representing or modeling graphs, such as degree distribution, effective diameter, clustering coefficient, and many more (Chakrabarti and Faloutsos, 2006; Drobyshevskiy and Turdakov, 2020).

Representing graphs by graph laws can be summarized into the following steps: (1) determine the parameter (or formula of several parameters) to represent the graphs, (2) fit the value of parameters based on the graph structures and features through statistical procedures. For example, Leskovec et al. (2005) discover the densification law over evolving graphs in the macroscopic view, which is expressed as $e(t) \propto n(t)^{\alpha}$, and $e(t)$ denotes the number of edges at time $t$, $n(t)$ denotes the number of nodes at time $t$, and $\alpha \in [1, 2]$ is an exponent parameter representing the density degree. And they use the empirical observation of real-world graphs to fit the value of $\alpha$. Targeting the microscopic view, Leskovec et al. (2008) discover other graph laws. Different from the macroscopic view, they view temporal graphs in a three-fold process, i.e., node arrival (determining how many nodes will be added), edge initiation (how many edges will be added), and edge destination (where are the added edges), where they ignore the deletion of nodes and edges. Then, they assign variables and corresponding equations (i.e., models) to parameterize these three processes and use MLE (i.e., maximum likelihood estimation) to settle the model and scalar parameters based on real-world graph observation. As an instance, the edge destination (i.e., the probability for node $u$ connecting node $v$) is modeled as $last^{\tau}$ other than $deg^{\tau}$ for the LinkedIn network through MLE, where $deg^{\tau}$ means the connection probability is proportional to node $v$'s current degree $d_t(v)^{\tau}$. And $last^{\tau}$ means the probability is proportional to node $v$'s age since its last interaction $\delta_t(v)^{\tau}$, where $\tau$ is the parameter to be fit.

Discovering graph laws and fitting law corresponding parameters can also serve many graph mining tasks and real-world applications. For example, after a graph law is discovered, the follow-up action is to propose the corresponding graph generative model to test whether there exists a realizable graph generator could generate graphs while preserving the discovered law in terms of graph properties (Leskovec et al., 2005; Zang et al., 2018; Do et al., 2020; Kook et al., 2020; Leskovec et al., 2008; Park and Kim, 2018; Zeno et al., 2020). Recently, the triadic closure law on temporal graphs (i.e., two nodes that share a common neighbor directly tend to connect) has been discovered to contribute to the dynamic link prediction task (Wang et al., 2021b). For the questions in social network analysis, e.g., "What is Twitter?", Kwak et al. (2010) give the statistical answer in the form of parametric representation. For pre-training the language model, the values of the weighted word co-occurrence matrix (i.e., adjacency matrix) are necessary and highly depend on the parameters following the power law, e.g., in GloVe (Pennington et al., 2014), $X_{ij}$ denotes the number of times that word $j$ occurs in the context of word $i$, and it follows $X_{ij} = \frac{k}{(r_{ij})^{\alpha}}$, where $r_{ij}$ denotes the frequency rank of the word pair $i$ and $j$ in the whole corpus, and $k$ and $\alpha$ are constant parameters.

#### 2.2.3 Graph Visualization (Visual Representation)

**Third, graph visualization provides visual representation by plotting the graph directly**, which is more straightforward than graph embedding and graph law to some extent. Hence, one of the research goals in graph visualization is finding the appropriate layout for the complex networked data. To name a few: most graphs (e.g., a five-node complete graph) could not be plotted on the plane without edge crossings, then Chen et al. (2020a) give the solution about how to use a 3D torus to represent the graph and then flatten the torus onto the 2D plane with aesthetics and representation accuracy preserved; Also, in (Nobre et al., 2020), authors evaluate which layouts (e.g., node-link diagram or matrix) are suitable for representing attributed graphs for different graph mining tasks; Through crowd-sourced experiments, Yang et al. (2020a) study the tactile representation of graphs for low-vision people and discuss which one (e.g., text, matrix, or node-link diagram) could help them to understand the graph topology; When the graph is large (e.g., hundreds of thousands of nodes), it is hard to represent the internal structure, and Nassar et al. (2020) design the high-order view of graphs (i.e., construct $k$-clique weighted adjacency matrix) and then use t-SNE to get the two-dimensional coordinates from the weighted Laplacian matrix. Bringing time information to graph visualization started in the 1990s to deal with the scenario where the represented graph gets updated (Beck et al., 2014). The trend for visualizing dynamic (or temporal) graphs becomes popular, and different research goals emerge (Kerracher et al., 2014; Beck et al., 2017), like strengthening the domain-specific evolution for domain experts (Bach et al., 2015), showing the pandemic dissemination (Lacasa et al., 2008; Tsiotas and Magafas, 2020), explaining time-series data (e.g., response time to different questions) with graph visualization and graph law (Mira-Iglesias et al., 2019).

Plotting graphs into an appropriate layout is more challenging when it comes to complex evolving graphs. Hence, many dynamic graph visualization research works contribute their solutions from different angles. For example, for balancing the trade-off between temporal coherence and spatial coherence (i.e., preservation of structure at a certain timestamp), Leydesdorff and Schank (2008) use the multidimensional scaling (MDS) method. Inspired by that, Xu et al. (2013) design the dynamic multidimensional scaling (DMDS), and Rauber et al. (2016) design the dynamic t-SNE; In order to assign end-users the flexibility to view the different aspects of evolving graphs (e.g., time-level graph evolution or node-level temporal evolution), Bach et al. (2014) represent evolving graphs into user-rotating cubes; To highlight the temporal relation among graph snapshots, authors in (Bach et al., 2016) propose Time Curves to visualize the temporal similarly between two consecutively observed adjacency matrices; In (Lentz et al., 2012; Pfitzner et al., 2012), researchers find that paths in temporal networks may invalidate the transitive assumption, which means the paths from node $a$ to node $b$ and from node $b$ to node $c$ may not imply a transitive path from node $a$ via node $b$ to node $c$. Inspired by this observation and to further analyze the actual length of paths in temporal graphs, Scholtes (2017) transfer this problem into investigating the order (i.e., $k$) of graphs. To be specific, the order $k$ can be understood as the length of a path (i.e., $v_{i-k} \to \ldots \to v_{i-1} \to v_i$) and can be modeled by the high-order Markov Chain (i.e., $\mathbb{P}(v_i | v_{i-k} \to \ldots \to v_{i-1})$). And the order of temporal paths can be determined by thresholding the probability gain in the MLE model. A corresponding follow-up visualization work is proposed targeting the high-order temporal graphs (Perri and Scholtes, 2019), which first determines the order of a temporal network as discussed above, and then constructs intermediate supernodes for deriving the high-order temporal relationship between two nodes, finally plots this high-order temporal relationship into edges and adds them on a static graph layout.

### 2.3 Graph Neural Networks

To extract the hidden representation, graph neural network (GNN), as a powerful tool, provides a new idea different from the embedding methods like DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), and node2vec (Grover and Leskovec, 2016). One major difference between GNNs and those mentioned above is that GNNs could aggregate multi-hop node features to represent a node by stacking GNN layers. According to (Xu et al., 2019b), this mechanism is called information aggregation (or message-passing in some literature), which iteratively updates the representation vector of a node by aggregating the representation vectors from its neighbors. The general formula of GNNs can be expressed as follows.

$$\mathbf{a}_v^{(k)} = \mathit{AGGREGATE}^{(k)}(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}),\ \mathbf{h}_v^{(k)} = \mathit{COMBINE}^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}_v^{(k)}) \tag{1}$$

where $\mathbf{h}_v^{(k)}$ is the hidden representation vector of node $v$ at the $k$-th iteration (i.e., $k$-th layer), and $\mathbf{a}_v^{(k)}$ is the aggregation among hidden representation vectors of neighbors $\mathcal{N}(v)$ of node $v$ from the last iteration (i.e., layer). For example, the graph convolutional neural network (GCN) (Kipf and Welling, 2017) can be written in the above formulation by integrating the *AGGREGATE* and *COMBINE* as follows.

$$\mathbf{h}_v^{(k)} = \mathit{ReLU}(\mathbf{W}^{(k-1)} \cdot \mathit{MEAN}\{\mathbf{h}_u^{(k-1)}, \forall u \in \mathcal{N}(v) \cup \{v\}\}) \tag{2}$$

where $\mathbf{W}^{(k-1)}$ is a learnable weight matrix at the $(k-1)$-th layer, and the original equation of GCN is as follows.

$$\mathbf{H}^{(k)} = \mathit{ReLU}(\hat{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k-1)}) \tag{3}$$

where $\hat{\mathbf{A}}$ is the normalized adjacency matrix with self-loops, i.e., $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, and $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$.

Graph neural network is a complicated computational framework that integrates the neural networks from deep learning and non-Euclidean constraints from graph theory. Therefore, GNN research consists of many specific facets from both ends. For example,

- Neural Layer Architecture Design: Recurrent (Li et al., 2018; Hajiramezanali et al., 2019), Residual Connections (Chen et al., 2020b; Zheng et al., 2022), etc.
- Message Passing Schema: Spectral Convolution (Kipf and Welling, 2017), Spatial Convolution (Velickovic et al., 2018), Simplification (Wu et al., 2019; Klicpera et al., 2019), etc.
- Training Manner: Semi-Supervised Learning (Kipf and Welling, 2017), Self-Supervised Learning (Velickovic et al., 2019; You et al., 2020), etc.
- Sampling Strategy: Noises-Aware (Yang et al., 2020b), Efficiency and Generalization (Hu et al., 2020a), Fairness-Preserving (Kang et al., 2022), etc.
- Model Trustworthy: Attack and Defend (Zhu et al., 2019; Zhang and Zitnik, 2020), Black-Box Explanation (Ying et al., 2019; Luo et al., 2020; Vu and Thai, 2020), etc.
- many more . . .

Until now, we have introduced three aspects of graph research shown in Figure 1. Targeting each aspect, research in natural and artificial dynamics could contribute to performance improvements. The detailed related works are discussed in the next section, where we start by defining the natural and artificial dynamics in graphs, and then investigate how natural and artificial dynamics help graph research enhancements in each specific aspect.

## 3 NATURAL AND ARTIFICIAL DYNAMICS IN GRAPHS

**Natural dynamics in graphs** means that the input graph (to graph mining, graph representations, and graph neural networks) has the naturally evolving part(s), such as the evolving World Wide Web. Formally speaking, the naturally evolving part means that the topological structures or node (edge, subgraph, or graph) features and labels depend on time. To be specific, the evolving graph structures can be represented either in

- (1) continuous time (Kazemi et al., 2020) or streaming (Aggarwal and Subbian, 2014): an evolving graph can be modeled by an initial state $G$ with a set of timestamped events $O$, and each event can be node/edge addition/deletion; or
- (2) discrete time (Kazemi et al., 2020) or snapshots (Aggarwal and Subbian, 2014): an evolving graph can be modeled as a sequence of time-respecting snapshots $G^{(1)}, G^{(2)}, \ldots, G^{(T)}$, and each $G^{(t)}$ has its own node set $V^{(t)}$ and edge set $E^{(t)}$.

For these two modelings, the corresponding time-dependent features and labels can be represented in a time-series or a sequence of matrices such as $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(T)}$.

These two modeling methods have non-trivial complements. For example, continuous-time models rapid node/edge-level evolution, i.e., microscopic evolution (Leskovec et al., 2008), such as protein molecule interactions in a cell (Fu and He, 2021a); However, it could not represent the episodic and slowly-changing evolution patterns, which can be captured by discrete-time, i.e., macroscopic evolution (Leskovec et al., 2005), such like the periodical metabolic cycles in a cell (Fu and He, 2021a). Recently, different evolution patterns in a single graph are currently not jointly modeled for improving graph representation comprehensiveness, but some real-world evolving graphs naturally have both evolution patterns. For example, in (Fu and He, 2021a), each dynamic protein-protein interaction network has 36 continuous observations (i.e., 36 edge timestamps), every 12 observations compose a metabolic cycle (i.e., 3 snapshots), and each cycle reflects 25 mins in the real world. Inspired by this observation, a nascent work (Fu et al., 2022b) is recently proposed to jointly model different evolution patterns into the graph representation.

**Artificial dynamics in graphs** means that the graph research related elements (e.g., graph topology, graph stream, node/graph attributes/labels, GNNs gradients and neural architectures, etc.) are deliberately re-designed by end-users for boosting the task performance in certain metrics. For the re-designing, end-users can *change (e.g., filter, mask, drop, or augment) the existing elements* or *construct (i.e., from scratch) non-existing elements* to improve the performance (e.g., decision accuracy, model robustness and interpretation, etc.) than the original. To name a few, one example of artificial dynamics can be graph augmentation: DropEdge (Rong et al., 2020) is proposed to deal with the over-fitting of GNNs by randomly removing a certain amount of edges from the input graphs for each training epoch; DummyNode (Liu et al., 2022b) is proposed to add a dummy node to the directed input graph, which connects all existing $n$ nodes with $2n$ directed edges. The dummy node serves as a highway to extend the information aggregation in GNNs and contribute to capturing the global graph information, such that the graph classification accuracy by GNNs can be enhanced. In addition to the graph augmentation, other specific examples of artificial dynamics can be filtering unimportant coming sub-structures to save computations (Fu et al., 2020b), adding residual connections among GNNs layers to address vanishing gradients (Zheng et al., 2022), and perturbing the GNNs gradients for privacy protection (Yang et al., 2021).

As mentioned above, on the one hand, considering the natural dynamics could leverage temporal dependency to contribute to graph research in terms of but not limited to, fast computation (e.g., tracking from the past instead of computing from scratch), causality reasoning (e.g., previous states cause the current state), comprehensive decision (e.g., prediction based on historical behaviors); On the other hand, studying artificial dynamics could help a wide range of targets, such as machine learning effectiveness (e.g., robustness, de-overfitting, de-oversmoothing).

Investigating natural dynamics and investigating artificial dynamics not only have shared merits but also have exclusive advantages. For example, how to manipulate evolving graphs is still an opening question for many downstream task improvements. Thus, a spontaneous research question is to ask whether natural dynamics can be integrated with artificial dynamics, which aims to keep the shared merits and bring exclusive advantages to synergy complementation. Definitely, some pioneering works have been proposed to touch this area. To introduce them, throughout the paper, we use **natural + artificial dynamics** to denote the integrated investigation of natural dynamics and artificial dynamics in graph-related research and then present related works in this category.

Starting from the following subsections, we are ready to introduce recent related works about natural, artificial, and natural + artificial dynamics research in graph mining, graph representations, and graph neural networks, respectively.

## 3.1 Dynamics in Graph Mining

Graph mining is a general term that consists of various specific mining tasks on graphs. Classic graph mining tasks consist of node clustering (or graph partitioning), node/graph classification, and link prediction. Also, motivated by real world application scenarios, novel graph mining tasks are being proposed for research, such as graph generation, graph alignment, and many more. Facing various graph mining tasks, we discuss several graph mining tasks here and then introduce the corresponding related works of natural dynamics, artificial dynamics, and natural + artificial dynamics in each discussed task.

### 3.1.1 Natural Dynamics in Graph Mining

**Link Prediction**. The core of the link prediction task is to decide whether there should be a link between two entities in the graph. This graph mining task can directly serve the recommender system by modeling the user and items as nodes in their interaction graphs. The evidence to decide whether two nodes should be linked can be the current heuristics like node embedding similarity (Zhang and Chen, 2018; Zhu et al., 2021), and also the historical behaviors of entities can be added for a more comprehensive decision. For example, JODIE (Kumar et al., 2019) is a link prediction model proposed based on user-item temporal interaction bipartite graph, where a user-item interaction is modeled as $(u, i, t, \mathbf{f})$ that means an interaction happens between user $u$ and item $i$ at time $t$, and $\mathbf{f}$ is the input feature vector of that interaction. Given a user (or an item) has a sequence of historical interactions (i.e., a user interacts with different items at different timestamps), JODIE (Kumar et al., 2019) applies two mutually-recursive RNN structures (i.e., $RNN_U$ and $RNN_I$) to update the embedding for users and items as follows.

$$
\begin{aligned}
\mathbf{u}(t) &= \sigma(\mathbf{W}_1^u\,\mathbf{u}(t^-) + \mathbf{W}_2^u\,\mathbf{i}(t^-) + \mathbf{W}_3^u\,\mathbf{f} + \mathbf{W}_4^u\,\Delta_u), \text{ embedding unit of } RNN_U \\
\mathbf{i}(t) &= \sigma(\mathbf{W}_1^i\,\mathbf{i}(t^-) + \mathbf{W}_2^i\,\mathbf{u}(t^-) + \mathbf{W}_3^i\,\mathbf{f} + \mathbf{W}_4^i\,\Delta_i), \text{ embedding unit of } RNN_I
\end{aligned}
\tag{4}
$$

where $\mathbf{W}_1^u$, $\mathbf{W}_2^u$, $\mathbf{W}_3^u$, and $\mathbf{W}_4^u$ are four parameters of $RNN_U$. And $RNN_U$ and $RNN_I$ share the same intuitive logic. Suppose user $u$ interacts with item $i$ at time $t$ with the interaction feature $\mathbf{f}$, then the above equation $RNN_U$ updates the user embedding $\mathbf{u}(t)$ at time $t$ by involving the latest historical user and item behavior, where $\Delta_u$ denotes the time elapsed since user $u$'s previous interaction with any item, $\mathbf{u}(t^-)$ denotes the latest user embedding vector right before time $t$, and $\mathbf{i}(t^-)$ denotes the latest item embedding vector right before time $t$. Therefore, in JODIE, each user (or item) can have a sequence of embedding vectors, which is called its trajectory. And the user and item embeddings can be updated iteratively to the future. The training loss is designed for whether the future user (or item) embedding vectors can be predicted [3]. If the future embedding can be predicted (e.g., $u$ connects $i$ at $t$, and $\mathbf{i}(t)$ is predicted through

[3] The future embedding vector estimation for users and items is skipped here.

$\mathbf{u}(t^-)$ and $\mathbf{i}(t^-)$), then the user (or item) historical evolution pattern is supposed to be encoded. Thus, the trained model can be used to predict whether a user $u$ interacts with an item $i$ in the future.

**Graph Alignment**. Compared with classic graph mining tasks, graph alignment is a relatively novel graph mining task, aiming to find paired (i.e., similar) nodes across two graphs. The input graphs can be attributed (e.g., heterogeneous information networks or knowledge graphs), and the proximity to decide whether two nodes from two different graphs are paired or not can range from their attributes, their neighborhood information (e.g., neighbor nodes attributes, connected edges' attributes, induced subgraph topology), etc. (Zhang and Tong, 2016; Yan et al., 2021b; Zhou et al., 2021). When aligning two graphs in the real world, the inevitable problem is that the input graphs are evolving in terms of features and topological structures. To this end, Yan et al. (2021a) combine two graphs into one graph, and then propose the GNN-based fast computation graph alignment method instead of re-training the GNN from scratch for each update of the combined graph. Specifically, authors want to encode the topology-invariant node embedding by training a GNN model, then fine-tune this trained GNN model with updated local changes (e.g., added nodes and edges, updated node input features). Thus, to weaken the coupling between the graph topology (e.g., adjacency matrix $\mathbf{A}$) and the GNN parameter matrix (e.g., $\mathbf{W}^{(k)}$ at the $k$-th layer), authors select GCN (Kipf and Welling, 2017) as the backbone and change its information aggregation schema by introducing a topology-invariant mask gate $\mathcal{M}^{(k)}$ and a highway gate $\mathcal{T}^{(k)}$ as follows.

$$\begin{aligned}\mathbf{H}^{(k)} &= \sigma(\hat{\mathbf{A}}\mathcal{M}^{(k-1)}(\mathbf{H}^{(k-1)})\mathbf{W}^{(k-1)}) \\ \mathbf{H}^{(k)} &= \mathcal{T}^{(k-1)}(\mathbf{H}^{(k-1)}) \odot \mathbf{H}^{(k)} + (1 - \mathcal{T}^{(k-1)}(\mathbf{H}^{(k-1)})) \odot \mathbf{H}^{(k-1)}\end{aligned} \tag{5}$$

where $\odot$ denotes Hadamard product, topology-invariant mask gate $\mathcal{M}^{(k-1)}(\mathbf{H}^{(k-1)})$ equals to $\mathbf{H}^{(k-1)} \odot \sigma(\mathbf{W}_m^{(k-1)})$, highway gate $\mathcal{T}^{(k-1)}(\mathbf{H}^{(k-1)})$ is expressed as $\sigma(\mathcal{M}^{(k-1)}(\mathbf{H}^{(k-1)})\mathbf{W}_h^{(k-1)})$, and $\mathbf{W}_m^{(k-1)}$ and $\mathbf{W}_h^{(k-1)}$ are learnable parameters of $\mathcal{M}^{(k-1)}$ and $\mathcal{T}^{(k-1)}$. The training loss function depends on whether the embedding vectors of two paired nodes (i.e., positive samples) are close, and whether the embedding vectors of two not paired nodes (i.e., negative samples) are far away. With this trained GNN model, future updates can be regarded as additional training samples to fine-tune the model.

#### 3.1.2 Artificial Dynamics in Graph Mining

**Graph Secure Generation or Graph Anonymization**. Graph generation is the task that models the given graphs' distribution and then generates many more meaningful graphs, which could contribute to various applications (Bonifati et al., 2020). However, approximating the observed graph distributions as much as possible will induce a privacy-leak risk in the generated graphs. For example, a node's identity is highly likely to be exposed in the generated social network if its connections are mostly preserved, which means a degree-based node attacker will easily detect a vulnerability in the generated graph with some background knowledge (Wu et al., 2010). Therefore, graph secure generation or graph anonymization is significant to social security (Fu et al., 2022c).

To protect privacy during the graph generation, artificial dynamics can help by introducing the perturbations during the modeling (or learning) of graph distributions. However, adding this kind of artificial dynamics to protect graph privacy still serves for the static graph generation. How to add dynamics to evolving graphs to protect privacy is still an opening question.

For privacy-preserving static graph generation, current solutions can be roughly classified into two types. First, the artificial dynamics is directly performed on the observed topology to generate new graph data, to name a few,

- Randomize the adjacency by iteratively switching existing edges $\{(t, w)$ and $(u, v)\}$ with $\{(t, v)$ and $(u, w)\}$ (if $(t, v)$ and $(u, w)$ do not exist in the original graph $G$), under the eigendecomposition preservation (Ying and Wu, 2008).
- Inject the connection uncertainty by iteratively copying each existing edge from original graph $G$ to a initial null graph $G'$ with a certain probability, guaranteeing the degree distribution of $G'$ is unchanged compared with $G$ (Nguyen et al., 2015).
- Permute the connection distribution by proportionally flipping the edges (existing to non-existing and vice versa), maintaining the edge-level differential privacy (edge-DP) for the graph structural preservation (Qin et al., 2017).

Second, following the synergy of deep learning and differential privacy (Abadi et al., 2016), another way to add artificial dynamics is targeting the gradient of deep graph learning models. To be specific, a deep graph generative model is recently proposed under privacy constraints, i.e., in (Yang et al., 2021), privacy protection mechanism is executed during the gradient descent phase of the generation learning process, by adding Gaussian noise to the gradient.

In terms of how to design appropriate artificial dynamics for the evolving graph secure generation, it is still a challenging problem because of maintaining privacy guarantee and utility preservation simultaneously. Here we would like to share our thoughts that the next-generation techniques should address the following challenges, at least.

- Unlike static graphs, what kind of natural dynamic information is sensitive in evolving graphs and should be hidden in the generated graph to protect entities' privacy is not clear.
- After the sensitive information is determined, the protection mechanism in the evolving environment is not yet available, e.g., dealing with changing topology and features.
- When the corresponding protection mechanism is designed, it can still be challenging to maintain the generation utility at the same time with privacy constraints.

### 3.1.3 Natural + Artificial Dynamics in Graph Mining

As mentioned in the above subsection, not only for the graph secure generation, adding artificial dynamics to evolving graphs is still nascent in many graph mining tasks, and exists many research opportunities. Here, we introduce a recent work that adds artificial dynamics to the time-evolving graph partitioning to improve computation efficiency.

**Node Clustering or Graph Partitioning**. In the node clustering family, local clustering methods target a specific seed node (or nodes) and obtain the clustering by searching the neighborhood instead of the entire graph. In this paper (Fu et al., 2020b), authors propose the motif-preserving local clustering method on temporal graphs called L-MEGA, which approximately tracks the local cluster position at each timestamp instead of solving it from scratch. To make L-MEGA more efficient, one speedup technique is

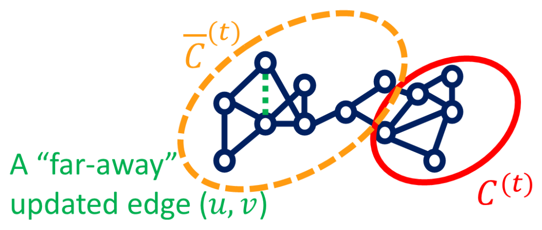


**Figure 2.** Local cluster $C^{(t)}$ and a "far-away" edge to be filtered at time $t$.

proposed in (Fu et al., 2020b) to filter the new arrival edges instead of letting them go into the tracking process and save them for future timestamps, if the new arrival edges are "far-away" from the current local cluster and do not affect the local structure as shown in Figure 2. By doing which, the tracking time complexity can be saved. In order to investigate whether a new arrival edge can be filtered, the authors identify the "far-away" edges by analyzing its incident nodes in terms of the probability mass in the personal PageRank vector and the shortest path to the local cluster.

## 3.2 Dynamics in Graph Representations

In this section, we mainly discuss graph embedding (i.e., graph representation learning) as one instance of graph representations, and introduce related works about how natural dynamics and artificial dynamics are involved in boosting the performance of graph representation learning[4].

### 3.2.1 Natural Dynamics in Graph Representations

In the early stage, inspired by DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), and node2vec (Grover and Leskovec, 2016), the graph embedding methods for temporal graphs are proposed, like CDTNE (Nguyen et al., 2018), DyGEM (Goyal et al., 2018), DynamicTriad (Goyal et al., 2018), HTNE (Zuo et al., 2018), FiGTNE (Liu et al., 2020), and tdGraphEmbd (Beladev et al., 2020). They vary in different ways to deal with time information. For example, FiGTNE (Liu et al., 2020) utilizes the temporal random walk to sample time-adjacent nodes. In this sampled sequence, the embedding is regularized such that previous nodes should reflect the current node.

Recently, inspired by GNNs stacking layers to aggregate multi-hop neighbor information to get node embedding vectors, temporal graph neural networks (TGNNs) are proposed to consider time information when doing the information aggregation, like EvolveGCN (Pareja et al., 2020), TGAT (Xu et al., 2020), and many others. In some works, they are also called spatial-temporal graph neural networks (STGNNs) because the spatial information comes from the input graph topological structure (Wu et al., 2021). In this paper, we use the term temporal graph neural networks, i.e., TGNNs, and the detailed related works for TGNNs are introduced in Section 3.3.1, i.e., *Natural Dynamics in Graph Neural Networks*.

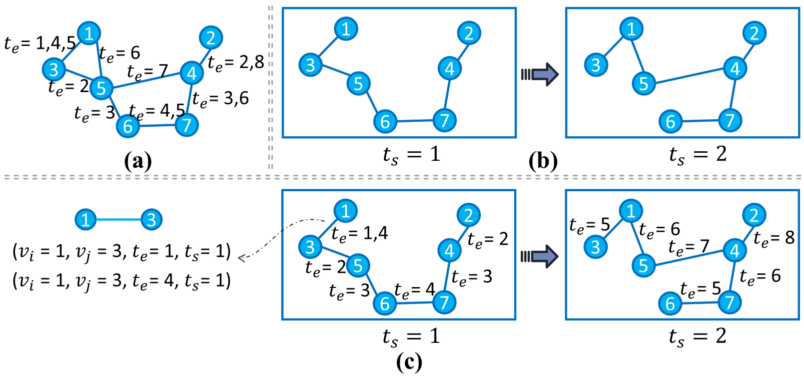


**Figure 3.** Part (a) shows a streaming graph with only edge timestamps $t_e$. Part (b) shows a snapshot-modeled graph with only snapshot timestamps $t_s$, where each $t_s$ elapses every 4 $t_e$. Part (c) shows our multi-time evolution modeling with edge timestamps $t_s$ and snapshot timestamps $t_e$.

**Multiple Evolution Patterns in Representation Learning**. As discussed earlier, in the real world, an evolving graph may have multiple evolution patterns (Fu and He, 2021a). Therefore, how to integrate multiple evolution patterns jointly during the representation learning process is still a nascent problem. Generally speaking, if we

[4] Here, we select graph embedding (i.e., graph representation learning) as an instance of graph representations to introduce the corresponding natural and artificial dynamic techniques. Since GNN is a also kind of tool for graph representation learning, then in this Section 3.2, we introduce the dynamic techniques that can be applied to general graph representation learning models. In Section 3.3, for GNNs, we will introduce the dynamic techniques that are deliberately designed for GNNs, which may or may not be applied to the general graph embedding models like DeepWalk, LINE, node2vec, etc.

model each evolution pattern as a different view of the input graph, then VANE (Fu et al., 2020a) could get the node embedding that is suitable for each observed view. Specifically, Temp-GFSM (Fu et al., 2022b) is proposed, which deliberately targets the streaming pattern for rapid node/edge-level evolution and the snapshot pattern for episodic and slowly-changing evolution, as shown in Figure 3. In Temp-GFSM, a multi-time attention mechanism is introduced with the support of the time kernel function to get the node-level, snapshot-level, and graph-level embeddings across different evolution patterns.

### 3.2.2 Artificial Dynamics in Graph Representations

**Pre-training for Representation Learning with Masked Graph Signals**. Generally speaking, training graph representation learning models (e.g., GNNs) is usually executed in the (semi-)supervised setting that requires a considerable amount of labeled data, especially when the input graphs are large. However, in some domains (e.g., healthcare (Choi et al., 2017)), collecting high-quality labeled graph data is usually time-consuming and costly. Therefore, recent advances have focused on the GNN pre-training (Hu et al., 2020b,c; Qiu et al., 2020; Li et al., 2021; Xu et al., 2021; Zhou et al., 2022), which pre-trains GNN models on the source domain(s) via proxy graph signals and then transfers pre-trained GNNs to the target domain. One common way of realizing proxy graph signal learning is to mask the input graphs in the unit of graph signals and train the GNNs such that they can predict the masked signals from the unmasked part. The masked signals range from masked node/edge/subgraph attributes and masked topology (e.g., nodes and edges) (Hu et al., 2020b,c). The quality of pre-trained GNNs can largely rely on (1) the relevance between the source domain(s) and the target domain and (2) the selection of masked graph signals, which may cause the negative transfer (Rosenstein et al., 2005) if (1) the source domain distribution diverges from the target domain distribution (i.e., cross-graph heterogeneity) or masked graph signals contradict each other (i.e., graph-signal heterogeneity) (Zhou et al., 2022). Inspired by that, Zhou et al. (2022) propose the MentorGNN to realize the domain-adaptive graph pre-training. To address the cross-graph heterogeneity, MentorGNN utilizes the multi-scale encoder-decoder architecture, such that knowledge transfer can be done in a coarser resolution (i.e., transfer the encoded source domain knowledge and decode it in the target domain) instead of being directly translated. The intuition behind this is that it is more common for different domain graphs to share high-level knowledge than very detailed knowledge. To address the graph-signal heterogeneity, MentorGNN dynamically re-weighting the importance of different kinds of masked graph signals via the curriculum learning framework in terms of the target domain performance.

### 3.2.3 Natural + Artificial Dynamics in Graph Representations

**Inserting Masks to Preserve Evolution Patterns during Temporal Graph Representation Learning**. Compared with baseline methods designed for static graph representation learning, considering the temporal information is more challenging and requires more consideration, like how to capture the evolution patterns of input graphs. In DySAT (Sankar et al., 2020), besides using structural attention like GAT (Velickovic et al., 2018) in each observed snapshot, authors design the temporal self-attention to get the node representation sequence from the first timestamp to the last timestamp, i.e., $\mathbf{z}_v = \{\mathbf{z}_v^{(1)}, \mathbf{z}_v^{(2)}, \ldots, \mathbf{z}_v^{(T)}\}$, for node $v$ at each observed timestamp. To preserve the evolution patterns when encoding $\mathbf{z}_v$, authors design the mask matrix

$\mathbf{M}$ as follows.

$$\begin{aligned} \mathbf{Z}_v =& \mathbf{B}_v(\mathbf{X}_v\mathbf{W}_v), \;\; \mathbf{B}_v(i,j) = \frac{exp(e_v^{ij})}{\sum_{k=1}^{T} exp(e_v^{ik})} \\ e_v^{ij} =& (\frac{(\mathbf{X}_v\mathbf{W}_q)(\mathbf{X}_v\mathbf{W}_k)_{ij}^{\top}}{\sqrt{F}} + \mathbf{M}(i,j)), \; i,j \in \{1,\dots,T\} \end{aligned} \tag{6}$$

where matrices $\mathbf{W}_q \in \mathbb{R}^{D\times F}$, $\mathbf{W}_k \in \mathbb{R}^{D\times F}$, and $\mathbf{W}_v \in \mathbb{R}^{D\times F}$ are query, key, value matrices in the standard self-attention mechanism (Vaswani et al., 2017). $\mathbf{X}_v \in \mathbb{R}^{T\times D}$ is the node feature of node $v$ across all $T$ timestamps, and $\mathbf{Z}_v \in \mathbb{R}^{T\times F}$ is the output time-aware representation matrix of node $v$. And $e_v^{ij}$ is the attention weight of timestamp $i$ to timestamp $j$ for node $v$, which is obtained through the mask matrix $\mathbf{M} \in \mathbb{R}^{T\times T}$.

$$\mathbf{M}(i,j) = \begin{cases} 0, & i \leq j \\ -\infty, & \text{otherwise} \end{cases} \tag{7}$$

The introduction of $\mathbf{M}$ preserves the evolution pattern in an auto-regressive manner. To be specific, when $\mathbf{M}(i,j) = -\infty$, the softmax attention weight $\mathbf{B}_v(i,j) = 0$, which turns off the attention weight from timestamp $i$ to timestamp $j$.

## 3.3 Dynamics in Graph Neural Networks

In this section, we focus on a specific kind of graph representation learning tool, graph neural network (GNN), and see how natural dynamics and artificial dynamics work in GNNs [5].

### 3.3.1 Natural Dynamics in Graph Neural Networks

**Temporal Graph Neural Networks (TGNNs)**. For temporal graph neural networks (TGNNs), the general principle is that the input graphs are evolving, e.g., the graph structure or node attributes are dependent on time. Since TGNNs take the graphs as input and the topological information is also called spatial information in some applications like traffic modeling (Yu et al., 2018a; Li et al., 2018), TGNNs are also called spatial-temporal graph neural networks (STGNNs or ST-GNNs) in some works (Wu et al., 2021). Here, we use the term temporal graph neural networks (TGNNs). How to deal with time information appropriately during the vanilla GNNs' information aggregation process is the key idea for TGNNs. Different works propose different manners, not limited to the following list.

- CNN-based TGNNs: In (Yan et al., 2018; Yu et al., 2018a), authors apply the convolutional operations from convolutional neural networks (CNNs) on graphs' evolving features to capture time-aware node hidden representations.
- RNN-based TGNNs: In (Li et al., 2018; Hajiramezanali et al., 2019; Pareja et al., 2020), authors inserts the recurrent units (from various RNNs such like LSTM and GRU) into GNNs to preserve the temporal dependency during the GNNs' representation learning process.
- Time Attention-based TGNNs: In (Sankar et al., 2020), authors propose using the self-attention mechanism on time features to learn the temporal correlations along with node representations..
- Time Point Process-based TGNNs: In (Trivedi et al., 2019), authors utilize Time Point Process to capture the interleaved dynamics and get time features.

[5] As mentioned earlier, in this Section 3.3 we introduce the natural and artificial dynamic techniques that are deliberately designed for GNNs, which may or may not be applied to the general graph embedding models.

- Time Kernel-based TGNNS: In (Xu et al., 2020), authors use Time Kernel to project time to a differential domain for the time representation vectors.

Let's take TGAT (Xu et al., 2020) as an instance of TGNNs, to illustrate the mechanism of encoding the temporal information into the node representations. TGAT uses the Time Kernel function $\mathbb{K}$ to project every observed time interval of node connections into a continuous differentiable functional domain, i.e., $\mathbb{K} : [t - \Delta t, t] \to \mathbb{R}^d$, in order to represent the time feature during the information aggregation mechanism of GNNs. Since TGAT is inspired by the self-attention mechanism (Vaswani et al., 2017), another benefit of introducing the Time Kernel is that the projected hidden representation vector can serve as the positional encoding in the self-attention mechanism. Time Kernel $\mathbb{K}$ can be realized by different specific functions (Xu et al., 2019a). For example, in TGAT (Xu et al., 2020),

$$\mathbb{K}(t_e - \Delta t, t_e) = \Psi(t_e - (t_e - \Delta t)) = \Psi(\Delta t) \tag{8}$$

and

$$\Psi(\Delta t) = \sqrt{\frac{1}{d}}[\cos \omega_1(\Delta t), \cos \omega_2(\Delta t), \ldots, \cos \omega_d(\Delta t)] \tag{9}$$

where $\Delta t = t_e - (t_e - \Delta t)$ denotes the input time interval, and $\{\omega_1, \ldots, \omega_d\}$ are learnable parameters.

With the above time encoding, TGAT can learn node representation $\mathbf{h}_v^{(t)}$ for node $v$ at time $t$ through a self-attention-like mechanism. Especially, TGAT sets node $v$ as the query node to query and aggregate attention weights from its one-hop time-aware neighbors, $\mathcal{N}_v^{(t)}$, to get $\mathbf{h}_v^{(t)}$. In $\mathcal{N}_v^{(t)}$, for each neighbor node $v'$, its node feature is the combination of the original input feature with the time kernel feature, i.e, $[\mathbf{x}_{v'} \| \mathbb{K}(t', t)] \in \mathbb{R}^{(m+d)}$, where $\mathbf{x}_{v'} \in \mathbb{R}^m$ is the original input feature of node $v'$, $\mathbb{K}(t', t) \in \mathbb{R}^d$ is the encoded temporal feature, and $t'$ is the time when node $v'$ and $v$ connects.

#### 3.3.2 Artificial Dynamics in Graph Neural Networks

**Graph Augmentation for GNNs**. One straightforward example to show artificial dynamics in graph neural networks is the graph augmentation designed for GNNs. In general, drop operations can also be considered a kind of augmentation operation (Rong et al., 2020). Because dropping parts of the input graph can make a new input graph, such that the volume and diversity of input graphs increase. In this viewpoint, at least, graph augmentation for GNNs can be categorized into three items.

- Only drop operation: In (Rong et al., 2020), authors propose DropEdge to drop a certain amount of edges in the input graphs before each epoch of GNN training, to alleviate the over-fitting problem of GNNs. Similar operations also include DropNode (Feng et al., 2020).
- Only add operation: In (Gilmer et al., 2017), authors propose to add a master node to connect all existing nodes in the input graph, which operation could serve as a global scratch for the message passing schema and transfer long distance information, to boost the molecule graph prediction.
- Refine operation: In (Jin et al., 2020), authors consider the problem setting given the input graph is not perfect (e.g., the adjacency matrix is poisoning attacked by adversarial edges). To be specific, they aim to investigate the low-rank property and feature smoothness to refine (i.e., not restricted to only adding or dropping) the original input graph and obtain the satisfied node classification accuracy.

More detailed operations like those mentioned above can be found in (Ding et al., 2022), where these augmentation operations can also be further categorized into learnable actions and random actions.

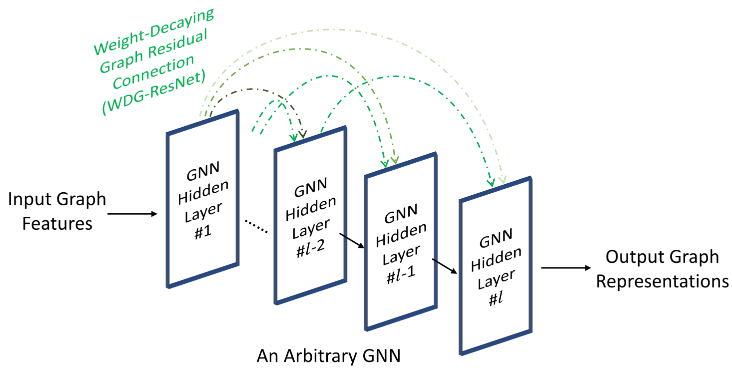


**Figure 4.** Adding Weight-Decaying Residual Connections on an Arbitrary GNN Architecture

**Adding Residual Connections among GNN layers**. When the input graph is imperfect (Xu et al., 2022) (e.g., topology and features are not consistent, features are partially missing), stacking more layers in GNNs can aggregate information from more neighbors to make the hidden representation more informative and serve various graph mining tasks (Zheng et al., 2022). However, the vanishing gradient problem hinders the neural networks from being deeper by making it hard-to-train, i.e., both the training error and test error of deeper neural networks are higher than shallow ones (He et al., 2016). The vanishing gradient problem can be illustrated as the gradients of the first few layers vanish, such that the training loss cannot be successfully propagated through deeper models. Currently, nascent deeper GNN methods (Zhao and Akoglu, 2020; Rong et al., 2020; Li et al., 2019) solve this problem by adding residual connections (i.e., ResNet (He et al., 2016)) on vanilla graph neural networks. In a recent study (Zheng et al., 2022), authors find that ResNet ignores the non-IID property of graphs, and directly adding ResNet on deeper GNNs will cause the shading neighbors effect. This effect distorts the topology information by making faraway neighbor information more important in deeper GNNs, such that it adds noise to the hidden representation and degrades the downstream task performance.

To address the shading neighbors effect, Zheng et al. (2022) design the weight-decaying graph residual connection (i.e., WDG-ResNet) deliberately for GNNs, as shown in Figure 4, which is expressed as follows.

$$
\begin{aligned}
\tilde{\mathbf{H}}^{(k)} &= \sigma(\hat{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k-1)}), \quad \text{/*}l\text{-th layer of an arbitrary GNN, e.g., GCN*/} \\
\mathbf{H}^{(k)} &= sim(\mathbf{H}^{(1)}, \tilde{\mathbf{H}}^{(k)}) \cdot e^{-k/\lambda} \cdot \tilde{\mathbf{H}}^{(k)} + \mathbf{H}^{(k-2)}, \quad \text{/*residual connection*/} \\
&= e^{cos(\mathbf{H}^{(1)}, \tilde{\mathbf{H}}^{(k)}) \; - \; k/\lambda} \cdot \tilde{\mathbf{H}}^{(k)} + \mathbf{H}^{(k-2)}
\end{aligned}
\tag{10}
$$

where $cos(\mathbf{H}^{(1)}, \tilde{\mathbf{H}}^{(k)}) = \frac{1}{n}\sum_i \frac{\mathbf{H}_i^{(1)}(\tilde{\mathbf{H}}_i^{(l)})^\top}{\|\mathbf{H}_i^{(1)}\|\|\tilde{\mathbf{H}}_i^{(l)}\|}$ measures the similarity between the $k$-th layer and the 1-st layer, and $\mathbf{H}_i^{(1)}$ is the hidden representation of node $i$ at the 1-st layer. The term $e^{-l/\lambda}$ is the decaying factor to further adjust the similarity weight of $\tilde{\mathbf{H}}^{(l)}$, where $\lambda$ is a constant hyperparameter. Compared to the vanilla ResNet (He et al., 2016), the WDG-ResNet introduces the decaying factor to preserve the hierarchical information of input graphs when the GNNs go deeper to alleviate the shading neighbors effect. Moreover, the authors empirically show that the optimal decaying factor is close to the diameter of input graphs, and such heuristics reduce the search space for hyperparameter optimization.

### 3.3.3 Natural + Artificial Dynamics in Graph Neural Networks

**Augmenting Temporal Graphs for TGNNs**. Augmenting evolving graphs has considerable research potential but has not attracted much attention yet (Ding et al., 2022). MeTA (Wang et al., 2021a) proposes an adaptive data augmentation approach for improving temporal graph representation learning using TGNNs. The core idea is modeling the realistic noise and adding the simulated noise to the low-information area of graphs (e.g., long time and far neighbors), in order to decrease the noise uniqueness for de-overfitting and increase the generalization ability of temporal graph representation learning process, to finally help downstream tasks such as link prediction. In (Wang et al., 2021a), authors propose three augmentation strategies: (1) perturbing time by adding Gaussian noise; (2) removing edges with a constant probability; (3) adding edges (i.e., sampled from the original graph) with perturbed time.

Research about augmenting temporal graphs is still in the nascent stage. And we would like to share, at least, the following research directions.

- Data-driven and learnable augmentation strategies for temporal graphs.
- Bounded augmentation solutions on temporal graphs, i.e., evolution patterns of original graphs can be preserved to some extent.
- Transferable and generalizable augmentation techniques across different temporal graphs.

## 4 DISCUSSION AND SUMMARY

In this paper, we first disentangle the graph-based research into three aspects (i.e., graph mining, graph representations, and graph neural networks) and then introduce the definition of natural and artificial dynamics in graphs. After that, we introduce related works in each combination between {graph mining, graph representations, and graph neural networks} and {natural dynamics, artificial dynamics, and natural + artificial dynamics}. In general, the topic of natural + artificial dynamics (i.e., adding artificial dynamics to evolving graphs) is still open in many graph research areas like graph mining, graph representations, and graph neural networks, and we list several opportunities in each corresponding subsection above. All opinions are authors' own and to the best of their knowledge. Also, due to the time limitation, many outstanding works are not discussed in this paper. We hope this paper can provide insights to relevant researchers and contribute to the graph research community.

## ACKNOWLEDGMENTS

This work is supported by National Science Foundation under Award No. IIS-1947203, IIS-2117902, and IIS-2137468. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

## REFERENCES

Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., et al. (2016). Deep learning with differential privacy. In SIGSAC 2016

Aggarwal, C. C. and Subbian, K. (2014). Evolutionary network analysis: A survey. ACM Comput. Surv.

Akoglu, L., Tong, H., and Koutra, D. (2015). Graph based anomaly detection and description: a survey. Data Min. Knowl. Discov.

Andersen, R., Chung, F. R. K., and Lang, K. J. (2006). Local graph partitioning using pagerank vectors. In FOCS 2006

Bach, B., Pietriga, E., and Fekete, J. (2014). Visualizing dynamic networks with matrix cubes. In CHI 2014

Bach, B., Riche, N. H., Fernandez, R., Giannisakis, E., Lee, B., and Fekete, J.-D. (2015). Networkcube: bringing dynamic network visualizations to domain scientists. In InfoVis 2015

Bach, B., Shi, C., Heulot, N., Madhyastha, T. M., Grabowski, T. J., and Dragicevic, P. (2016). Time curves: Folding time to visualize patterns of temporal evolution in data. IEEE Trans. Vis. Comput. Graph.

Beck, F., Burch, M., Diehl, S., and Weiskopf, D. (2014). The state of the art in visualizing dynamic graphs. In EuroVis 2014

Beck, F., Burch, M., Diehl, S., and Weiskopf, D. (2017). A taxonomy and survey of dynamic graph visualization. Comput. Graph. Forum

Beladev, M., Rokach, L., Katz, G., Guy, I., and Radinsky, K. (2020). tdgraphembed: Temporal dynamic graph-level embedding. In CIKM 2020

Bianchi, F. M., Grattarola, D., and Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling. In ICML 2020

Bojchevski, A., Shchur, O., Zügner, D., and Günnemann, S. (2018). Netgan: Generating graphs via random walks. In ICML 2018

Bonifati, A., Holubová, I., Prat-Pérez, A., and Sakr, S. (2020). Graph generators: State of the art and open challenges. ACM Comput. Surv.

Chakrabarti, D. and Faloutsos, C. (2006). Graph mining: Laws, generators, and algorithms. ACM Comput. Surv.

Chen, K., Dwyer, T., Marriott, K., and Bach, B. (2020a). Doughnets: Visualising networks using torus wrapping. In CHI 2020

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. (2020b). Simple and deep graph convolutional networks. In ICML 2020

Chiang, W., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In KDD 2019

Choi, E., Bahadori, M. T., Song, L., Stewart, W. F., and Sun, J. (2017). GRAM: graph-based attention model for healthcare representation learning. In KDD 2017

Ding, K., Xu, Z., Tong, H., and Liu, H. (2022). Data augmentation for deep graph learning: A survey. CoRR

Do, M. T., Yoon, S., Hooi, B., and Shin, K. (2020). Structural patterns and generative models of real-world hypergraphs. In KDD 2020

Dong, Y., Chawla, N. V., and Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. In KDD 2017

Drobyshevskiy, M. and Turdakov, D. (2020). Random graph modeling: A survey of the concepts. ACM Comput. Surv.

Du, B., Zhang, S., Cao, N., and Tong, H. (2017). FIRST: fast interactive attributed subgraph matching. In KDD 2017

Dunlavy, D. M., Kolda, T. G., and Acar, E. (2011). Temporal link prediction using matrix and tensor factorizations. ACM Trans. Knowl. Discov. Data

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, Y. E., Tang, J., et al. (2019). Graph neural networks for social recommendation. In WWW 2019

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., et al. (2020). Graph random neural networks for semi-supervised learning on graphs. In NeurIPS 2020

Fu, D., Ban, Y., Tong, H., Maciejewski, R., and He, J. (2022a). Disco: Comprehensive and explainable disinformation detection. In CIKM 2022

Fu, D., Fang, L., Maciejewski, R., Torvik, V. I., and He, J. (2022b). Meta-learned metrics over multi-evolution temporal graphs. In KDD 2022

Fu, D. and He, J. (2021a). DPPIN: A biological repository of dynamic protein-protein interaction network data. CoRR

Fu, D. and He, J. (2021b). SDG: A simplified and dynamic graph neural network. In SIGIR 2021

Fu, D., He, J., Tong, H., and Maciejewski, R. (2022c). Privacy-preserving graph analytics: Secure generation and federated learning. CoRR

Fu, D., Xu, Z., Li, B., Tong, H., and He, J. (2020a). A view-adversarial framework for multi-view network embedding. In CIKM 2020

Fu, D., Zhou, D., and He, J. (2020b). Local motif clustering on time-evolving graphs. In KDD 2020

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In ICML 2017

Goyal, P., Kamra, N., He, X., and Liu, Y. (2018). Dyngem: Deep embedding method for dynamic graphs. CoRR

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In KDD 2016

Hajiramezanali, E., Hasanzadeh, A., Narayanan, K. R., Duffield, N., Zhou, M., and Qian, X. (2019). In NeurIPS 2019

Hamilton, W. L., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In NeurIPS 2017

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In CVPR 2016

Hu, S., Xiong, Z., Qu, M., Yuan, X., Côté, M., Liu, Z., et al. (2020a). Graph policy network for transferable active learning on graphs. In NeurIPS 2020

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V. S., et al. (2020b). Strategies for pre-training graph neural networks. In ICLR 2020

Hu, Z., Dong, Y., Wang, K., Chang, K., and Sun, Y. (2020c). GPT-GNN: generative pre-training of graph neural networks. In KDD 2020

Jin, W., Barzilay, R., and Jaakkola, T. S. (2018). Junction tree variational autoencoder for molecular graph generation. In ICML 2018

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. (2020). Graph structure learning for robust graph neural networks. In KDD 2020

Jing, B., Park, C., and Tong, H. (2021). HDMI: high-order deep multiplex infomax. In WWW 2021

Kamvar, S. D., Haveliwala, T. H., Manning, C. D., and Golub, G. H. (2003). Extrapolation methods for accelerating pagerank computations. In WWW 2003

Kang, J., Zhou, Q., and Tong, H. (2022). Jurygcn: Quantifying jackknife uncertainty on graph convolutional networks. In KDD 2022

Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., et al. (2020). Representation learning for dynamic graphs: A survey. J. Mach. Learn. Res.

Kerracher, N., Kennedy, J., and Chalmers, K. (2014). The design space of temporal graph visualisation. In EuroVis 2014

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In ICLR 2017

Klicpera, J., Bojchevski, A., and Günnemann, S. (2019). Predict then propagate: Graph neural networks meet personalized pagerank. In ICLR 2019

Kook, Y., Ko, J., and Shin, K. (2020). Evolution of real-world hypergraphs: Patterns and models without oracles. In ICDM 2020

Kumar, S., Zhang, X., and Leskovec, J. (2019). Predicting dynamic embedding trajectory in temporal interaction networks. In KDD 2019

Kwak, H., Lee, C., Park, H., and Moon, S. B. (2010). What is twitter, a social network or a news media? In WWW 2010

Lacasa, L., Luque, B., Ballesteros, F., Luque, J., and Nuno, J. C. (2008). From time series to complex networks: The visibility graph. PNAS

Lentz, H., Selhorst, T., and Sokolov, I. M. (2012). Unfolding accessibility provides a macroscopic approach to temporal networks. CoRR

Leskovec, J., Backstrom, L., Kumar, R., and Tomkins, A. (2008). Microscopic evolution of social networks. In KDD 2008

Leskovec, J. and Faloutsos, C. (2007). Scalable modeling of real graphs using kronecker multiplication. In ICML 2007

Leskovec, J., Kleinberg, J. M., and Faloutsos, C. (2005). Graphs over time: densification laws, shrinking diameters and possible explanations. In KDD 2005

Leydesdorff, L. and Schank, T. (2008). Dynamic animations of journal maps: Indicators of structural changes and interdisciplinary developments. J. Assoc. Inf. Sci. Technol.

Li, G., Müller, M., Thabet, A. K., and Ghanem, B. (2019). Deepgcns: Can gcns go as deep as cnns? In ICCV 2019

Li, H., Wang, X., Zhang, Z., Yuan, Z., Li, H., and Zhu, W. (2021). Disentangled contrastive learning on graphs. In NeurIPS 2021

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2018). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In ICLR 2018

Liu, L., Du, B., Ji, H., Zhai, C., and Tong, H. (2021). Neural-answering logical queries on knowledge graphs. In KDD 2021

Liu, M., Luo, Y., Uchino, K., Maruhashi, K., and Ji, S. (2022a). Generating 3d molecules for target protein binding. In ICML 2022

Liu, Q., Allamanis, M., Brockschmidt, M., and Gaunt, A. L. (2018). Constrained graph variational autoencoders for molecule design. In NeurIPS 2018

Liu, X., Cheng, J., Song, Y., and Jiang, X. (2022b). Boosting graph structure learning with dummy nodes. In ICML 2022

Liu, Z., Zhou, D., Zhu, Y., Gu, J., and He, J. (2020). Towards fine-grained temporal network representation via time-reinforced random walk. In AAAI 2020

Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., et al. (2020). Parameterized explainer for graph neural network. In NeurIPS 2020

Luo, Y. and Ji, S. (2022). An autoregressive flow model for 3d molecular geometry generation from scratch. In ICLR 2022

Mira-Iglesias, A., Navarro-Pardo, E., and Conejero, J. A. (2019). Power-law distribution of natural visibility graphs from reaction times series. Symmetry

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. CoRR

Nassar, H., Kennedy, C., Jain, S., Benson, A. R., and Gleich, D. F. (2020). Using cliques with higher-order spectral embeddings improves graph visualizations. In WWW 2020

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In NeurIPS 2001

Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., and Kim, S. (2018). Continuous-time dynamic network embeddings. In Companion of WWW 2018

Nguyen, H. H., Imine, A., and Rusinowitch, M. (2015). Anonymizing social graphs via uncertainty semantics. In CCS 2015

Nguyen, V., Sugiyama, K., Nakov, P., and Kan, M. (2020). FANG: leveraging social context for fake news detection using graph representation. In CIKM 2020

Nobre, C., Wootton, D., Harrison, L., and Lex, A. (2020). Evaluating multivariate network visualization techniques using a validated design and crowdsourcing approach. In CHI 2020

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., et al. (2020). Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In AAAI 2020

Park, H. and Kim, M. (2018). Evograph: An effective and efficient graph upscaling method for preserving graph properties. In KDD 2018

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In EMNLP 2014

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: online learning of social representations. In KDD 2014

Perri, V. and Scholtes, I. (2019). Higher-order visualization of causal structures in dynamics graphs. CoRR

Pfitzner, R., Scholtes, I., Garas, A., Tessone, C. J., and Schweitzer, F. (2012). Betweenness preference: Quantifying correlations in the topological dynamics of temporal networks. CoRR

Qin, Z., Yu, T., Yang, Y., Khalil, I., Xiao, X., and Ren, K. (2017). Generating synthetic decentralized social graphs with local differential privacy. In CCS 2017

Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., et al. (2020). GCC: graph contrastive coding for graph neural network pre-training. In KDD 2020

Rauber, P. E., Falcão, A. X., and Telea, A. C. (2016). Visualizing time-dependent data using dynamic t-sne. In EuroVis 2016

Rong, Y., Huang, W., Xu, T., and Huang, J. (2020). Dropedge: Towards deep graph convolutional networks on node classification. In ICLR 2020

Rosenstein, M. T., Marx, Z., Kaelbling, L. P., and Dietterich, T. G. (2005). To transfer or not to transfer. In NIPS 2005 workshop on transfer learning. vol. 898, 1–4

Sankar, A., Wu, Y., Gou, L., Zhang, W., and Yang, H. (2020). Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In WSDM 2020

Saxena, A., Chakrabarti, S., and Talukdar, P. P. (2021). Question answering over temporal knowledge graphs. In ACL 2021

Scholtes, I. (2017). When is a network a network?: Multi-order graphical model selection in pathways and temporal networks. In KDD 2017

Shang, C., Wang, G., Qi, P., and Huang, J. (2022). Improving time sensitivity for question answering over temporal knowledge graphs. In ACL 2022

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell.

Simonovsky, M. and Komodakis, N. (2018). Graphvae: Towards generation of small graphs using variational autoencoders. CoRR

Spielman, D. A. and Teng, S. (2013). A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. SIAM J. Comput.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). LINE: large-scale information network embedding. In WWW 2015

Tong, H., Faloutsos, C., Gallagher, B., and Eliassi-Rad, T. (2007). Fast best-effort pattern matching in large attributed graphs. In KDD 2007

Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. (2019). Dyrep: Learning representations over dynamic graphs. In ICLR 2019

Tsiotas, D. and Magafas, L. (2020). The effect of anti-covid-19 policies on the evolution of the disease: A complex network analysis of the successful case of greece. Physics

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In NeurIPS 2017

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In ICLR 2018

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep graph infomax. In ICLR 2019

Vu, M. N. and Thai, M. T. (2020). Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. In NeurIPS 2020

Wang, D., Qi, Y., Lin, J., Cui, P., Jia, Q., Wang, Z., et al. (2019). A semi-supervised graph attentive network for financial fraud detection. In ICDM 2019

Wang, Y., Cai, Y., Liang, Y., Ding, H., Wang, C., Bhatia, S., et al. (2021a). Adaptive data augmentation on temporal graphs. In NeurIPS 2021

Wang, Y., Chang, Y., Liu, Y., Leskovec, J., and Li, P. (2021b). Inductive representation learning in temporal networks via causal anonymous walks. In ICLR 2021

Wu, F., Jr., A. H. S., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Q. (2019). Simplifying graph convolutional networks. In ICML 2019

Wu, X., Ying, X., Liu, K., and Chen, L. (2010). A survey of privacy-preservation of graphs and social networks. In Managing and Mining Graph Data

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst.

Xu, D., Cheng, W., Luo, D., Chen, H., and Zhang, X. (2021). Infogcl: Information-aware graph contrastive learning. In NeurIPS 2021

Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., and Achan, K. (2019a). Self-attention with functional time representation learning. In NeurIPS 2019

Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., and Achan, K. (2020). Inductive representation learning on temporal graphs. In ICLR 2020

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019b). How powerful are graph neural networks? In ICLR 2019

Xu, K. S., Kliger, M., and III, A. O. H. (2013). A regularized graph layout framework for dynamic network visualization. Data Min. Knowl. Discov.

Xu, Z., Du, B., and Tong, H. (2022). Graph sanitation with application to node classification. In WWW 2022

Yan, S., Xiong, Y., and Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. In AAAI 2018

Yan, Y., Liu, L., Ban, Y., Jing, B., and Tong, H. (2021a). Dynamic knowledge graph alignment. In AAAI 2021

Yan, Y., Zhang, S., and Tong, H. (2021b). BRIGHT: A bridging algorithm for network alignment. In WWW 2021

Yang, C., Wang, H., Zhang, K., Chen, L., and Sun, L. (2021). Secure deep graph generation with link differential privacy. In IJCAI 2021

Yang, Y., Marriott, K., Butler, M., Goncu, C., and Holloway, L. (2020a). Tactile presentation of network data: Text, matrix or diagram? In CHI 2020

Yang, Z., Ding, M., Zhou, C., Yang, H., Zhou, J., and Tang, J. (2020b). Understanding negative sampling in graph representation learning. In KDD 2020

Ying, X. and Wu, X. (2008). Randomizing social networks: a spectrum preserving approach. In SDM 2008

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019). Gnnexplainer: Generating explanations for graph neural networks. In NeurIPS 2019

You, J., Ying, R., Ren, X., Hamilton, W. L., and Leskovec, J. (2018). Graphrnn: Generating realistic graphs with deep auto-regressive models. In ICML 2018

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020). Graph contrastive learning with augmentations. In NeurIPS 2020

Yu, B., Yin, H., and Zhu, Z. (2018a). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In IJCAI 2018

Yu, W., Cheng, W., Aggarwal, C. C., Zhang, K., Chen, H., and Wang, W. (2018b). Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In KDD 2018

Zang, C., Cui, P., Faloutsos, C., and Zhu, W. (2018). On power law growth of social networks. IEEE Trans. Knowl. Data Eng.

Zeno, G., Fond, T. L., and Neville, J. (2020). Dynamic network modeling from motif-activity. In WWW 2020

Zhang, M. and Chen, Y. (2018). Link prediction based on graph neural networks. In NeurIPS 2018

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018). An end-to-end deep learning architecture for graph classification. In AAAI 2018

Zhang, S., Li, S., and Yang, J. (2009). GADDI: distance index based subgraph matching in biological networks. In EDBT 2009

Zhang, S. and Tong, H. (2016). FINAL: fast attributed network alignment. In KDD 2016

Zhang, X. and Zitnik, M. (2020). Gnnguard: Defending graph neural networks against adversarial attacks. In NeurIPS 2020

Zhao, L. and Akoglu, L. (2020). Pairnorm: Tackling oversmoothing in gnns. In ICLR 2020

Zheng, L., Fu, D., Maciejewski, R., and He, J. (2022). Deeper-gxx: Deepening arbitrary gnns. CoRR

Zheng, L., Li, Z., Li, J., Li, Z., and Gao, J. (2019). Addgraph: Anomaly detection in dynamic graph using attention-based temporal GCN. In IJCAI 2019

Zhou, D., Zheng, L., Fu, D., Han, J., and He, J. (2022). Mentorgnn: Deriving curriculum for pre-training gnns. In CIKM 2022

Zhou, D., Zheng, L., Han, J., and He, J. (2020). A data-driven graph generative model for temporal interaction networks. In KDD 2020

Zhou, D., Zheng, L., Xu, J., and He, J. (2019). Misc-gan: A multi-scale generative model for graphs. Frontiers Big Data

Zhou, Q., Li, L., Wu, X., Cao, N., Ying, L., and Tong, H. (2021). Attent: Active attributed network alignment. In WWW 2021

Zhu, D., Zhang, Z., Cui, P., and Zhu, W. (2019). Robust graph convolutional networks against adversarial attacks. In KDD 2019

Zhu, Z., Zhang, Z., Xhonneux, L. A. C., and Tang, J. (2021). Neural bellman-ford networks: A general graph neural network framework for link prediction. In NeurIPS 2021

Zuo, Y., Liu, G., Lin, H., Guo, J., Hu, X., and Wu, J. (2018). Embedding temporal network via neighborhood formation. In KDD 2018