A Comprehensive Survey on Graph Summarization with Graph Neural Networks

Nasrin Shabani, Jia Wu *Senior Member, IEEE*, Amin Beheshti, Quan Z. Sheng
Jin Foo, Venus Haghighi, Ambreen Hanif, Maryam Shahabikargar

Abstract—As large-scale graphs become more widespread, more and more computational challenges with extracting, processing, and interpreting large graph data are being exposed. It is therefore natural to search for ways to summarize these expansive graphs while preserving their key characteristics. In the past, most graph summarization techniques sought to capture the most important part of a graph statistically. However, today, the high dimensionality and complexity of modern graph data are making deep learning techniques more popular. Hence, this paper presents a comprehensive survey of progress in deep learning summarization techniques that rely on graph neural networks (GNNs). Our investigation includes a review of the current state-of-the-art approaches, including recurrent GNNs, convolutional GNNs, graph autoencoders, and graph attention networks. A new burgeoning line of research is also discussed where graph reinforcement learning is being used to evaluate and improve the quality of graph summaries. Additionally, the survey provides details of benchmark datasets, evaluation metrics, and open-source tools that are often employed in experimentation settings, along with a detailed comparison, discussion, and takeaways for the research community focused on graph summarization. Finally, the survey concludes with a number of open research challenges to motivate further study in this area.

Impact Statement—Graph summarization is a key task in managing large graphs, which are ubiquitous in modern applications. In this article, we summarize the latest developments in graph summarization methods, offer a more profound understanding of these methods, and list source codes and available resources. The study covers a broad range of techniques, including both conventional and deep learning-based approaches, with a particular emphasis on GNNs. We aim to help the researchers develop a basic understanding of GNN-based methods for graph summarization, benefit from useful resources, and think about future directions.

Index Terms—Deep Learning, Graph Neural Networks, Graph Summarization

I. INTRODUCTION

LARGE graphs are becoming increasingly ubiquitous. With the increasing amount of data being generated, large graphs are becoming more prevalent in modelling a variety of domains, such as social networks, proteins, the World Wide Web, user actions, and beyond. However, as these graphs grow in size, understanding and analyzing them is becoming

more challenging. Additionally, performing fast computations with large graphs and visualizing the knowledge they can yield is also becoming more difficult. Many claim that faster and more effective algorithms are needed to overcome these obstacles [1], [2]. However, a growing cohort of researchers believe that summarization might hold the answer to this unyielding problem. Summarization not only helps existing algorithms to parse the data faster, it can also compress the data, reduce storage requirements, and assist with graph visualization and sense-making [3].

Graph summarization is the process of finding a condensed representation of a graph while preserving its key properties [2]. A toy example of a typical graph summarization process is shown in Figure 1. The process includes removing the original graph's objects and replacing them with fewer objects of the same type to produce a condensed representation of the original graph.

Most traditional approaches to graph summarization involve using a conventional machine learning method or a graph-structured query, such as degree, adjacency, or eigenvector centrality, to find a condensed graphical representation of the graph [2]. A popular summarization technique is to group structures in the input graph by aggregating the densest subgraphs [4]. For example, the GraSS model [5] focuses on accurate query handling and incorporates formal semantics for answering queries on graph structure summaries based on a random walk model, while Graph Cube [6] is a data warehousing model that integrates both network structure summarization and attribute aggregation. This model also supports OLAP queries on large multidimensional networks.

Notably, clustering methods follow a similar approach to summarization, partitioning a graph into groups of nodes that can be further summarized. Most traditional graph clustering methods use conventional machine learning and statistical inference to measure the closeness of nodes based on their connectivity and structural similarities [7]. For instance, Karrer et al. [8] used a stochastic block model to detect clusters or communities in large sparse graphs. However, another method of graph summarization focuses more on node selection and identifying sparse graphs that can be used to derive smaller graphs [9]. As an example, Doerr et al. [10] introduced a sampling method based on traversing a graph that begins with a collection of starting points, e.g., nodes or edges, and then adds to the sample pool depending on recent information about the graph objects. However, despite the popularity of these approaches in the past, they are very computationally-intensive. They also require a great deal of memory to store,
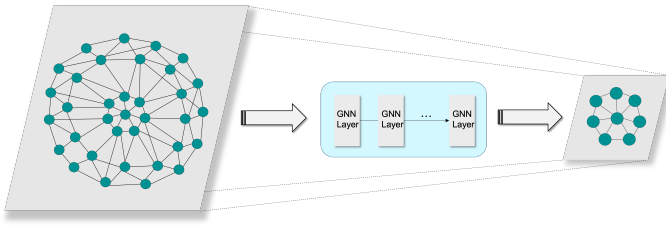
Fig. 1. An example of a graph summarization process. Objects are removed from the original graph and replaced with fewer objects of the same type to result in a condensed representation of the original graph.

making them unsuitable for today's modern, complex, and large-scale datasets.

Deep learning on graphs is a powerful computational technique for graphs of all sizes that continues to interest both academics and industry. Graph neural networks (GNNs) are the most successful paradigm among the deep learning techniques for graphs. Their multilayer deep neural networks are not only able to reduce the dimensionality of tasks, they also support high-speed calculations with large-scale graphs [11]. Several strategies that use a GNN as a summarization engine have been proposed over the last few years, and this line of research is expected to generate even more fruitful results in the future. These GNN-based methods have demonstrated promising performance with a diverse range of tasks, such as graph classification, node representation learning, and link prediction. Further, as research in this area continues to evolve, we can anticipate even more innovative and effective approaches to graph summarization that rely on GNNs.

A. Existing Surveys on Graph Summarization

Over the past decade, numerous reviews have been conducted that acknowledge the importance of graph summarization. They generally cover a range of topics, like graph partitioning, graph sampling, graph coarsening, and graph embedding, along with specific use cases for graph summarization, such as pattern discovery, community detection, fraud detection, and link prediction. Additionally, several comprehensive surveys on graph summarization techniques have been conducted based on scientific computing: [2], [3], [12]. Yet only the most current work by Chen et al. [12] compares the most recent machine learning techniques to traditional methods.

There are also several surveys on graph sampling [13], [14], graph partitioning [15], [16], and graph clustering [17], [18]. Other surveys focus on graph representation learning [19], [20] and graph embedding [21], [22]. As these streams of research look to create low-dimension versions of a graph, they are strongly connected to the concept of graph summarization. However, although these studies provide in-depth analyses of how graph summarization techniques are being used in important and high-demand domains, GNN-based graph summarization methods are not their main area of focus. Consequently, these surveys do not provide a comprehensive and structured evaluation of all available techniques for graph summarization.

B. Contributions

In this survey, we review developments in graph summarization with GNNs. Overall, we have aimed to provide researchers and practitioners with a comprehensive understanding of past, present, and future approaches to graph summarization using GNNs. Our specific contributions include:

- **The first deep GNN-based graph summarization survey.** To our best knowledge, this paper is the first thorough survey that is devoted to graph summarization with GNNs. Previous surveys have primarily concentrated on traditional graph summarization methods without considering deep learning techniques. As there is no existing specialized study on graph summarization with GNNs, this research aims to fill that gap and facilitate progress in the field through a detailed and structured survey.
- **Comprehensive review.** We present a comprehensive review of GNN-based graph summarization methods. We also highlight the benefits of different GNN techniques compared to traditional methods. Within each method, we present a review of the most notable approaches and their contributions to the field.
- **Resources and reproducing existing results.** We have put together a set of resources that support graph summarization with GNNs, including cutting-edge models, standardized datasets for comparison, and reproducing existing publicly available implementations. Our goal is to provide a resource for those seeking to understand and apply GNNs for graph summarization.
- **Future directions.** We identify and discuss the open challenges and new directions that lie ahead. Through an exploration of the existing challenges and potential avenues for progress, we aim to guide future research and development in GNNs for graph summarization.

The majority of the papers we reviewed were published at prominent international conferences (e.g., SIGKDD, NeurIPS, ICLR, ICML, KDD, WWW, IJCAI, and VLDB) or in peer-reviewed journals (e.g, ACM, IEEE, Elsevier, and Springer) in the domains of artificial intelligence, big data, data mining, and web services. In our investigations, we found that different fields referred to graph summarization using different terms, e.g., "coarsening", "reduction", "simplification", "abstraction", and "compression". While these concepts were used relatively interchangeably, the terms coarsening, simplification, and summarization were generally more common.

The remaining sections of this survey are structured as follows: Section II introduces the concept of graph summarization with primary definitions and background information. Section III provides an overview of the development of graph summarization. Section IV outlines the recently developed GNN-based graph summarization methods. Section V discusses graph reinforcement learning methods for graph summarization. Section VI outlines the structure of widely adopted implementation resources. Finally, Section VII explores a number of open research challenges that would motivate further study before the conclusion in Section VIII.

II. DEFINITIONS AND BACKGROUND

This section provides an overview of the key definitions and background information on graph summarization techniques.

- *Definition 1 (Graph)*: Graph G can be represented as a tuple (V, E), where V denotes the set of nodes or vertices $\{v_1, v_2, ..., v_n\}$, and E represents the set of edges or links $E = \{e_{ij}\}_{i,j=1}^n$ connecting node pairs. The graph is represented by an $n \times n$ dimensional adjacency matrix $A = [a_{ij}]$, with a_{ij} being 1 if the edge e_{ij} is present in E and 0 otherwise. If a_{ij} is not equal to a_{ji}, the graph is directed; otherwise, it is undirected. When edges are associated with weights from the set W, the graph is called a weighted network; otherwise, it is an unweighted network. G is considered labeled if every edge $e \in E$ has an associated label. Additionally, if each node $v \in V$ has a unique label, the nodes are also labeled; otherwise, G is considered unlabelled.
- *Definition 2 (Graph Summary)*: Given a graph G, a summary $G(V_S, E_s)$ is a condensed representation of G that preserves its key properties. Graph summarization techniques involve either aggregation, selection, or transformation on a given graph and produce a graph summary as the output.

As outlined in Definition 2, graph summarization approaches fall into three main categories: aggregation, selection, and transformation. While selection approaches make graphs sparser by simply removing objects without replacing them, aggregation approaches replace those removed objects with similar objects only with fewer of them. For example, a supernode might replace a group of nodes. Similar to selection and aggregation, the transformation approaches also involve removing objects from the graph, but this time the objects removed are transformed into a different type of object, such as an embedding vector [23].

Aggregation. Aggregation is one of the most extensively employed techniques of graph summarization. Aggregation methods can be divided into two main groups: those that involve node grouping and those that involve edge grouping. Node grouping methods group nodes into supernodes, whereas edge grouping methods reduce the number of edges in a graph by aggregating them into virtual nodes. Clustering and community detection are examples of a grouping-based approach. Although summarizing graphs is not explicitly the primary objective of these processes, the outputs can be modified into non-application-specific summaries [2].

Selection. There are two main groups of selection techniques: sampling and simplification. While sampling methods focus on picking subsets of nodes and edges from the input graph [24], simplification or sparsification methods involve removing less important edges or nodes. In this way, they tend to resemble solutions to dimensionality reduction problems [9].

Transformation. Graph projection and graph embedding are two categories of this method. Generally, graph projection refers to the summarization techniques that transform bipartite graphs with various inter-layer nodes and edges into simple (single-layer) summarized graphs. Conversely, graph embedding refers to the techniques that transform a graph into a lower dimensional representation while preserving the original graph's topology [25].

III. GRAPH SUMMARIZATION: AN EVOLUTION

Graph summarization has been playing an important role in areas such as network analysis, data mining, machine learning, and visualization for some time. The evolution of graph summarization is illustrated in Figure 2, which shows how it has progressed from traditional computing methods to multi-layer GNNs. This section briefly overviews the three different traditional methods within this field and explains the advantages of GNN techniques over traditional ones.

A. Clustering-based Approaches

Graph clustering can be thought of as a graph summarization technique since it involves grouping together nodes in a graph that are similar or related and, in so doing, the complexity and size of the original graph are reduced. In simpler terms, graph clustering provides a way to compress or summarize a large and complex graph into a smaller set of clusters, each of which captures some aspect of the structure or function of the original graph [1]. Graph summarization techniques using clustering can be classified into three main categories: structural-based, attribute-based, and structural-attribute-based approaches. The latter, combining both structural and attribute information, is considered the most effective [26]. For example, Boobalan et al. [27] proposed a method called k-Neighborhood Attribute Structural Similarity (k-NASS) that incorporates both structural and attribute similarities of graph nodes. This method improves clustering accuracy for complex graphs with rich attributes. However, clustering large graphs with many attributes remains challenging due to high memory and computational requirements.

B. Statistical Inference

Statistical inference techniques for graph summarization simplify the complexity of the original graph while preserving its significant characteristics. These techniques fall into two groups: pattern mining and sampling. Pattern mining identifies representative patterns or subgraphs in the graph to create a condensed summary. On the other hand, sampling randomly selects a subset of nodes or edges from the graph and estimates the properties of the entire graph based on this subset. One example of a sampling technique is Node2vec [28], which generates random sequences of nodes within a graph, known as walks, to create a graph summary. Various sampling techniques, such as random sampling [28], stratified sampling [29], and snowball sampling [30], can be used for graph summarization. Each technique has its advantages and disadvantages, and the choice depends on the specific problem and data being addressed.

C. Goal-driven

Goal-driven techniques for graph summarization involve constructing a graph summary that is tailored to a specific application or task. They are a powerful tool for capturing
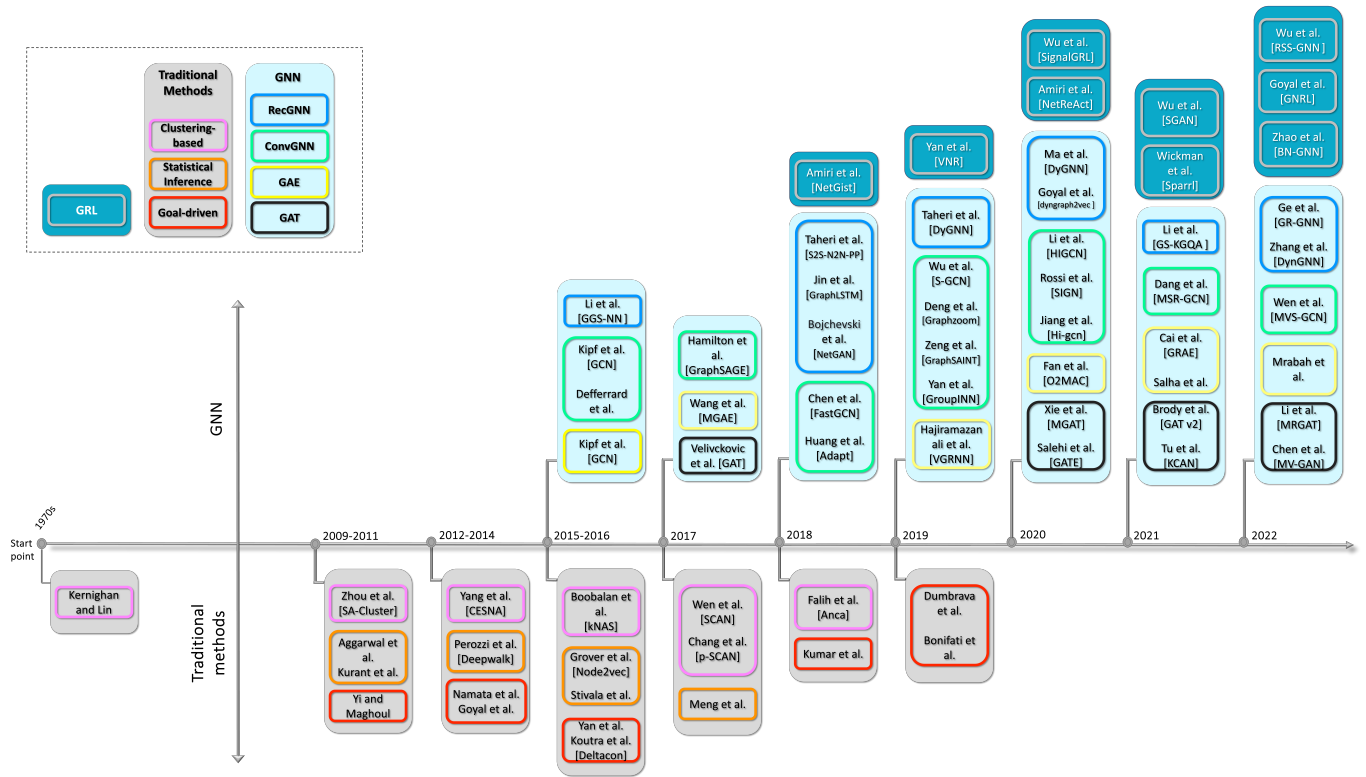
Fig. 2. A timeline of graph summarization and reviewed techniques.

specific features or relationships in a graph that are relevant to a specific application or task. By optimizing the graph summary to a specific goal, it is possible to create a more effective and efficient summary that can be used to derive better insights and make better decisions [31]. Significant goal-driven techniques for graph summarization include utility-driven and query-driven techniques. Utility-driven techniques aim to summarize large graphs while preserving their essential properties and structure to maximize their usefulness for downstream tasks. Human reviewers evaluate the utility of the summary against specific tasks like node classification and link prediction [32]. Query-driven techniques summarize graphs by identifying relevant subgraphs or patterns using queries in a query language. The resulting subgraph that matches the query becomes a building block for the graph summary, supporting the target downstream task [33]. The choice of the goal-driven summarization technique depends on the specific goals of the analysis, as some techniques may preserve global properties, while others may capture local structures. It also depends on available computational resources and the complexity and size of the original graph.

D. Why GNNs for Graph Summarization?

In recent times, deep learning has gained significant prominence and is now considered one of the most effective forms of AI due to its high accuracy. Conventional deep learning methods have shown that they perform extremely well with Euclidean data (e.g., images, signals, and text), and now there are a growing number of applications in non-Euclidean

domains (e.g., graphs and manifold structures). As a deep learning approach, GNNs are multi-layer neural networks that learn on graph structures to ultimately perform graph-related tasks like classification, clustering, pattern mining, and summarization [34].

As mentioned, traditional graph summarization approaches are mostly based on conventional machine learning or graph-structured queries, such as degree, adjacency, and eigenvector centrality, where the aim is to find a condensed graphical representation of the whole graph [6]. However, the pairwise similarity calculations involved in these approaches demand a considerably high level of computational power. The explicit learning capabilities of GNNs skirt this problem. Additionally, powerful models can be built from even low-dimensional representations of attributed graphs [36]. Unlike standard machine learning algorithms, with a GNN, there is no need to traverse all possible orders of the nodes to represent a graph. Instead, GNNs consider each node separately without taking the order of the input nodes into account. This avoids redundant computations.

The major advantage of GNN models for graph summarization over traditional methods is the ability to use low-dimensional vectors to represent the features of large graphs [37]. Additionally, the message-passing mechanism used by GNNs to communicate information from one node to the next has been the most successful learning framework for learning the patterns and neighbours of nodes and the sub-graphs in large graphs [11]. It is also easy to train a GNN in semi- or unsupervised way to aggregate, select, or
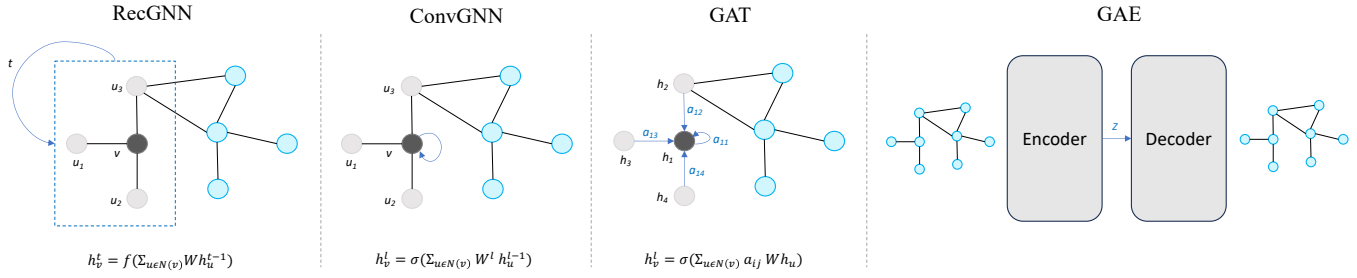
Fig. 3. GNN Architechtures [34], [35].

transform graphs into low dimensional representations [38]. In this regard, recent successes in graph summarization with GNNs point to promising directions for new research. For example, the GNN models developed by Brody et al. [39] and Goyal et al. [40] represent dynamic graphs in low dimensions, providing a good foundation for popularizing GNNs into more complex dynamic graphs.

IV. GRAPH SUMMARIZATION WITH GNNS

This section provides an overview of recent research into graph summarization with GNNs. Each subsection covers four main categories of approach – these being: Recurrent Graph Neural Networks (RecGNNs), Convolutional Graph Neural Networks (ConvGNNs), Graph Autoencoders (GAEs), and Graph Attention Networks (GATs). Four different types of GNN models are shown in Figure 3. Within each subsection, we first provide a brief introduction about the architecture of the GNN model and then review the most notable approaches and the contributions each has made to the field. At the end of each subsection, we provide a comprehensive summary of the key features of representative GNN-based approaches. We present a comparative analysis in Tables I, II, III, and IV for RecGNN, ConvGNN, GAE, and GAT architectures, respectively. The tables include comparisons of evaluation methods, performance metrics, training data, advantages, and limitations across the different models.

A. RecGNN-based Approaches

RecGNNs are early implementations of GNNs, designed to acquire node representations through a generalized recurrent neural network (RNN) architecture. Within these frameworks, information is transmitted between nodes and their neighbours to reach a stable equilibrium [34], [41]. A node's hidden state is continuously updated by

$$h_v^t = f\left(\sum_{u \in N(v)} W h_u^{t-1}\right) \quad (1)$$

where h_v^t is the hidden state of node v at time t, which represents the information learned by the RecGNN about node v at a specific time step in the dynamic graph sequence. $N(v)$ denotes the set of neighboring nodes of node v in the graph, providing the context and connectivity information for node v within the graph structure. h_u^{t-1} is the hidden state

of neighboring node u at time $t - 1$, which contributes to the update of node v's hidden state at the previous time step, reflecting the influence of neighboring nodes on v's representation. W is the weight matrix used for aggregating the hidden states of neighboring nodes.

Here, $f(\cdot)$ is usually a simple element-wise activation like ReLU, tanh, or sigmoid. This simple activation is typically used to introduce non-linearity and capture complex patterns in the node representations. However, this can be replaced by recurrent update functions, which use gated mechanisms like LSTM (Long Short-Term Memory) [42] or GRU (Gated Recurrent Units) [43] cells. In this case, each node's hidden state update would be computed using an LSTM or GRU cell, which is more complex and sophisticated than simple element-wise activation functions. These cells determine how the hidden state of node v at time t is updated based on its current and previous hidden states and the information from its neighboring nodes, enabling the model to capture temporal dependencies in the dynamic graph-structured data [44].

RecGNN-based approaches for graph summarization mostly focus on graph sampling and embedding by generating sequences from graphs and embedding those sequences into a continuous vector space at lower dimensions. In the following, we will first briefly introduce LSTM and GRU architectures and then delve into the graph summarization approaches that are built upon their respective structures.

1) LSTM-based Approaches: LSTMs are a class of RNNs that use a sequence of gates to concurrently model long- and short-term dependencies in sequential data [54]. The modified architecture of an LSTM to handle large graph-structured data is known as a GraphLSTM [55]. Typically, the input to the model consists of a sequence of either graph nodes or edges, which are processed in order using LSTM units. At each time step, the model updates its internal state based on the input node or edge and the previous state, as shown in Equation 2.

$$h_v^t = LSTM\left(\sum_{u \in N(v)} W h_u^{t-1}\right) \quad (2)$$

The cell is composed of multiple gates, and its operation can be described as follows [55]:

$$f_v^t = \sigma(W_f \times [h_v^{t-1}, x_v^t] + b_f) \quad (3)$$

$$i_v^t = \sigma(W_i [h_v^{t-1}, x_v^t] + b_i) \quad (4)$$

TABLE I
COMPARATIVE ANALYSIS OF SELECTED RECGNN-BASED APPROACHES FOR GRAPH SUMMARIZATION.

Ref.	Model Name	Approach	Evaluation	Performance Metrics	Training Data	Advantages	Limitations
Taheri et al. [45]	S2S-N2N-PP	LSTM	Node Classification	Accuracy	Labeled, Unlabeled	Robust performance, capturing global structure, unsupervised learning.	Limited performance with Weisfeiler-Lehman labels, sensitive to noise.
Jin et al. [46]	GraphLSTM	LSTM	Graph Classification	Accuracy	Labeled	Incorporation of long-term dependencies, effective node representations.	Dependency on pretrained embeddings, sensitivity to node ordering, lack of discussion on scalability.
Bojchevski et al. [47]	NetGAN	LSTM	Link Prediction	Link Prediction Accuracy	Labeled	Generative approach, scalability, unsupervised learning.	practical limitations with massive graphs, training instability due to use of GANs.
Wu et al. [48]	GSGAN	LSTM	Community Detection	ARI	Labeled	Flexibility in sparsification, handling large data by using random walks, accurate noise filtering.	Limited guarantee for analysis, adding artificial edges (contradicts with the graph summariziation's aim), lack of comparison with other models.
Ma et al. [49]	DyGNN	LSTM	Node Classification, Link Prediction	MRR, Recall	Labeled, Unlabeled	Incorporation of temporal information, consideration of varied influence.	Limited label information, no parameter analysis for hyper parameters.
Khoshraftar et al. [50]	LSTM-Node2vec	LSTM	Node Classification, Link Prediction, Anomaly Detection	AUC, F1-score	Labeled	Incorporates temporal information, outperforms state-of-the-art methods, preserves long-term dependencies.	Fixed length of history, Memory intensive, no attention mechanism.
Li et al. [51]	GGS-NNS	GRU	Node Classification	Accuracy	Labeled	Capturing temporal information, Reasonable scalability, end-to-end learning	Limited global information, complexity for large graphs, task-specific architecture.
Taheri et al. [52]	DyGGNN	GRU	Graph Classification	Accuracy	Labeled	Capturing topological and temporal features, Sequence-to-sequence architecture, unsupervised.	Not scalable to very large graphs, Limited evaluation, complexity and computational cost.
Ge et al. [53]	GR-GNN	GRU	Graph Classification, Node Classification	Micro F1-score	Labeled	Deep feature extraction, robustness and universality, computational efficiency.	Limited scope of evaluation, limited extraction of deep chain-dependent features.

$$o_v^t = \sigma(W_o[h_v^{t-1}, x_v^t] + b_o) \qquad (5)$$

$$C_v^t = tanh(W_C[h_v^{t-1}, x_v^t] + b_C) \qquad (6)$$

$$c_v^t = f_v^t \times c_v^{t-1} + i_v^t \times C_v^t \qquad (7)$$

In the given equations, the variables f_v^t, i_v^t, and o_v^t serve as the forget, input, and output gates, respectively. The forget gate plays a crucial role in determining what information to retain or discard from the cell state (long-term memory) for node v at time t. On the other hand, the input gate is responsible for determining which new information should be stored in the cell state for the node. Also, the output gate regulates what information is to be outputted from the current cell state. Finally, the hidden state for node v at time t is updated using the output gate and the cell state as follows:

$$h_v^t = o_v^t \times tanh(c_v^t) \qquad (8)$$

The cell state c_v^t represents the memory cell of the LSTM network at time t. It acts as a long-term memory, capable of storing information over extended sequences. To calculate c_v^t, the new candidate value C_v^t is first calculated and then the cell state for node v at time t is updated using the forget gate, input gate, and the new candidate values. The gates take into consideration the previous hidden state h_v^{t-1} of node v at time $t-1$ and the current input feature vector x_v^t. Moreover, W_x and b_x are the weights and biases, respectively, associated with each of the gates.

By repeating this process over multiple time steps, the model captures the dependencies in the graph and generates a final hidden state that summarizes all the information contained in the entire graph. This makes it possible to preserve a record of both the previous inputs and the structure of the graph. After processing all the nodes and edges, the final state of the graph LSTM is used as the summary representation of the graph. This summary vector can then be used as input to downstream machine learning models or as a feature for other graph analysis tasks [56].

LSTM-based approaches for graph summarization have proven to be effective with a range of tasks, such as graph clustering, graph classification, and link prediction. For example, Taheri et al. [45] leveraged various graph-LSTM-based methods to generate sequences from a graph structure, including breadth-first search, random walk, and shortest path. Finally, they trained a graph LSTM to embed those graph sequences into a continuous vector space. The hidden states at time step t of the encoder are as follows:

$$h_t^{enc} = LSTM_{enc}(x_t, h_{t-1}^{enc}) \qquad (9)$$

where x_t is the input vector at time t and h_t^{enc} denotes the hidden state at time step t in a given trained encoder $LSTM_{enc}$. Similarly, the hidden state at time step t of the decoder is defined in Equation 10.

$$h_t^{dec} = LSTM_{dec}(x_{t-1}, h_{t-1}^{dec}) \qquad (10)$$

Bojchevski et al. [47] proposed NetGAN, a recurrent-based model to capture the underlying structural information that leads to the learning of meaningful network embeddings. NetGAN leverages the power of generative models to learn a compact representation of a graph via random walks, enabling more efficient processing of large graphs while preserving their essential features. As a variant, Wu et al. [48] modified the NetGAN model to form a new social network with artificial edges that is suitable for community detection.

Jin et al. [46] also developed an approach to learning representations of graphs based on a graph LSTM. Here, graph representations of diverse sizes are encoded into low-dimensional vectors. Li et al. [57] proposed a graph summarization technique that uses a graph LSTM and a ConvGNN to improve question answering with knowledge graphs. In this approach, the questions, entities, and relations are represented as vectors with very few dimensions, but the key properties of the relations are well preserved.

Several studies have also focused on evolving node patterns in dynamic graphs. For instance, Zhang et al. [56] introduced an LSTM-based approach, a one-stage model called DynGNN. The model embeds an RNN into a GNN model to produce a representation in compact form. Khoshraftar et al. [50] presented a dynamic graph embedding method via LSTM to convert a large graph into a low-dimensional representation. The model captures temporal changes with LSTM using temporal walks and then transfers the learned parameters into node2vec [28] to incorporate the local structure of each graph. Similarly, Ma et al. [49] introduced a dynamic RecGNN model that relies on a graph LSTM to model the dynamic information in an evolving graph while reducing the graph's dimensionality and learning manifold structures. Node information is continuously updated by: recording the time intervals between edges; recording the sequence of edges; and coherently transmitting information between nodes. Another work by Goyal et al. [40] also presents a method for learning temporal transitions in dynamic graphs. This framework is based on a deep architecture that mainly consists of dense and recurrent layers. Model size and the number of weights to be trained can be a problem during training, but the authors overcome this issue with a uniform sampling of nodes.

2) GRU-based Approaches: GRUs are a variant of graph LSTMs that include a gated RNN structure and have fewer training parameters than a standard graph LSTM. The key distinction between a GRU and an LSTM is the number of gates in each model. GRU units are less complex with only two gates, "reset" and "update" [58].

$$r_v^t = \sigma(W_r[h_v^{t-1}, x_v^t] + b_r) \tag{11}$$

$$z_v^t = \sigma(W_z[h_v^{t-1}, x_v^t] + b_z) \tag{12}$$

$$C_v^t = tanh(W_C[r_v^t \times h_v^{t-1}, x_v^t] + b_C) \tag{13}$$

$$h_v^t = (1 - z_v^t) \times h_v^{t-1} + z_v^t \times C_v^t \tag{14}$$

In these equations, r_v^t is a reset gate and z_v^t is update gate. h_v^{t-1} is the output of the model at the previous time step. Similar to LSTM, C_v^t computes the new candidate value, and h_v^t updated hidden state for node v at time t using the update gate and the new candidate values. b_x, W_x are biases and weights for respective gates.

Again, by repeating this process over several time steps, the model learns the dependencies that exist between the nodes in the graph, allowing it to construct a final hidden state that summarizes all the graph's information. The adaptability of GRU cells to capture temporal dependencies within graph-structured data allows for effective information aggregation and context modeling, making GRU-based methods a promising choice for summarizing complex graph-structured information. For instance, Taheri et al. [52] proposed the DyGrAE model, which is able to learn the structure and temporal dynamics of a dynamic graph while condensing its dimensions. A GRU model captures the graph's topology, while an LSTM model learns the graph's dynamics. Ge et al. [53] developed a gated recursive algorithm that not only solves some node aggregation problems but also extracts deeply dependent features between nodes. The resulting model, called GR-GNN, is based on a GRU which performs the aggregation and structure. Li et al.'s GRU model [51] encodes an input graph into a fixed-size vector representation, which is then fed into a sequence decoder to generate the summary as the output. The model effectively captures the structural information and dependencies among the nodes and edges in the input graph, which is crucial for producing a coherent and informative graph summary.

B. ConvGNN-based Approaches

The general idea of ConvGNN-based approaches is to generalize the CNNs on graph-structured data [34]. The primary distinction between a ConvGNN and a RecGNN is the way information is propagated. While ConvGNNs apply various weights at each timestep, RecGNNs apply the same weight matrices in an iterative manner until an equilibrium is reached [69].

In other words, ConvGNN models are a form of neural network architecture that supports graph structures and aggregates node information from the neighbourhood of each node in a convolutional manner. ConvGNN models have demonstrated a strong expressive capacity for learning graph representations, resulting in superior performance with graph summarization [69].

ConvGNN-based approaches fall into two categories: spectral-based and spatial-based methods [34].

1) Spectral-based Approaches: Spectral-based methods describe graph convolutions based on spectral graph theory and graph signal filtering. In spectral graph theory, the multiplication of the graph with a filter (the convolution) is defined in a Fourier domain [70].

Although the computation contains well-defined translational properties, it is relatively expensive, and the filters are not generally localized. Since the level of complexity grows with the scale of the graphs, one solution is to only check a limited number of neighbours using Chebyshev's theory [71]. The Chebyshev polynomials $T_k(x)$ are defined recursively as:

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \tag{15}$$

where $T_0(x) = 1$ and $T_1(x) = x$. Here, x represents the variable of the Chebyshev polynomial. k is a non-negative integer that determines the degree of the polynomial which is the order of the Chebyshev polynomial. It is a polynomial function of degree k, and its value depends on the values of x and k as defined by the recurrence relation mentioned in Equation 15.

TABLE II
COMPARATIVE ANALYSIS OF SELECTED CONVGNN-BASED APPROACHES FOR GRAPH SUMMARIZATION.

Ref.	Model Name	Approach	Evaluation	Performance Metrics	Training Data	Advantages	Limitations
Kipf et al. [59]	GCN	Spectral	Node Classification	Accuracy	Labeled	Overcoming limitations of graph laplacian regularization, less complex and better scalability, improved predictive performance.	Memory requirement, limited to directed edges, does not support edge features, limiting assumptions.
Wu et al. [60]	SGC	Spectral	Node Classification	Accuracy, Micro F1-score	Labeled	Ease of implementation, applicability to large-scale graphs, memory efficiency.	Limited expressiveness, loss of hierarchical representations, limitations in handling complex graphs due to its simplified nature.
Deng et al. [61]	GraphZoom	Spectral	Node Classification	Accuracy, Micro F1-score	Labeled	Applicable to different types of graphs, preservation of local and global structure, scalable to huge graphs.	Lack of node label preservation, limited support for large graphs.
Rossi et al. [62]	SIGN	Spectral	Node Classification	Accuracy	Labeled	Robustness to graph irregularities, general applicability (can be applied to various graph-based tasks), faster training and inference.	Limited to undirected graphs, dependent on the choice of operator combinations.
Jiang et al. [63]	Hi-GCN	Spectral	Graph Classification	Accuracy, AUC, Sensitivity and Specificity	Labeled	Hierarchical graph convolution, contribution to neuroscience, consideration of correlation.	Complex optimization, limited comparison with prior works.
Hamilton et al. [64]	GraphSAGE	Spatial	Node Classification	Micro F1-score	Labeled	Scalability, inductive learning, flexibility in using aggregation strategies.	Assumption of homophily, limited global context, over-smoothing, hyperparameter sensitivity.
Chen et al. [65]	FastGCN	Spatial	Node Classification	Micro F1-score	Labeled, Unlabeled	Ease of sampling implementation, retains model accuracy despite using importance sampling to speed up training.	Limited comparison with the state-of-the-art, opportunities to reduce sampling variance remain.
Zeng et al. [66]	GraphSAINT	Spatial	Node Classification	Accuracy, Micro F1-score	Labeled	Efficient training on large graphs by introducing "neighbor explosion", low training complexity.	Sampling overhead, pre-processing requirements, limited generalization to unseen graphs.
Yan et al. [67]	GroupINN	Spatial	Node Classification	Accuracy, Micro F1-score	Labeled	Provides interpretability, parameter reduction, capturing complex relationships.	Limited application, dependency on data quality, lack of generalizability to different datasets or tasks outside the scope of the neuroscience domain.
Wen et al. [68]	MVS-GCN	Spatial	Graph Classification	Accuracy, AUC, Sensitivity and Specificity	Labeled	Multi-view brain network embedding, interpretability, effective graph structure learning.	Limited contribution of negative connectives, influence of hyperparameters, challenges in interpreting complex brain networks.

This theory has led to several studies that explore the idea of applying approximation to the spectral graph convolution. For example, Defferrard et al. [72] generalized the CNNs to graphs by designing localised convolutional filters on graphs. Their main idea was to leverage the spectral domain of graphs and use Chebyshev polynomial approximation to efficiently compute localized filters as follows:

$$Z = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{L}) X \qquad (16)$$

where L represents a graph Laplacian, X is the node feature matrix, θ is the graph convolutional operator with a filter parameter, $\widetilde{L}$ is a scaled Laplacian defined as $\widetilde{L} = 2L/I - I$, and K is the order of the Chebyshev polynomial approximation. The θ_k parameters are the learnable filter coefficients. They also introduced a graph summarization procedure that groups similar vertices together and a graph pooling procedure that focuses on producing a higher filter resolution. This work has been used as the basis of several studies that draw on Chebyshev polynomials to speed up convolution computations.

As a variant, Kipf et al. [59] introduced several simplifications to the original framework to improve the model's classification performance and scalability to large networks. These simplifications formed the basis of the GCN (Graph Convolutional Network) model, which has since become a popular choice for various graph-related tasks. Given an undirected graph with an adjacency matrix A, they computed the normalized graph Laplacian $\widetilde{L}$ as follows:

$$\widetilde{L} = I - D^{-1/2} A D^{-1/2} \qquad (17)$$

where I is the identity matrix, and D is the diagonal degree matrix, with D_{ii} representing the sum of the weights of the edges connected to node i.

In this work, instead of directly computing graph convolutions using high-order Chebyshev polynomials, as done in the previous work by Defferrard et al. [72], Kipf et al. proposed using a simple first-order approximation of graph filters. They defined the graph convolution operation as [59]:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-1/2} \widetilde{A} \widetilde{D}^{-1/2} H^{(l)} W^{(l)}) \qquad (18)$$

where $H^{(l)}$ represents the hidden node features at layer l. $W^{(l)}$ is the weight matrix for the layer l. $\widetilde{A} = A + I$ is the adjacency matrix with self-loops added. And, $\widetilde{D}$ is the diagonal degree matrix of $\widetilde{A}$. Here, the normalized graph Laplacian $\widetilde{D}^{-1/2} \widetilde{A} \widetilde{D}^{-1/2}$ is used to aggregate information from neighboring nodes. Hence, the propagation can be written as follows:

$$h_i^{(l+1)} = \sigma \left(\sum_{j \in \mathcal{N}_i} \frac{1}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}} \cdot h_j^{(l)} \cdot W^{(l)} \right) \quad (19)$$

where $h_i^{(l)}$ is the feature vector of node i at layer l. $\mathcal{N}_i$ is the set of neighbors of node i in the graph. $W^{(l)}$ is the weight matrix for the l-th layer. $\sigma(\cdot)$ is the activation function applied element-wise.

More recently, several successful variations of the spectral method have been proposed, e.g., S-GCN [60] and later SIGN [62]. S-GCN is a simplified GCN model but does not come with any performance compromises in terms of graph summarization. The idea behind the S-GCN model is to first convert large convolutional filters into smaller ones and then remove the final non-linear layers. Inspired by previous ConvGNN models, Rossi et al. [62] subsequently proposed SIGN, which scales ConvGNNs to extremely large graphs by combining various amendable graph convolutional filters for faster training and sampling purposes.

Another prominent line of research in spectral-based ConvGNN approaches revolves around transforming graph objects, e.g., embedding. For example, Jiang et al. [63] introduced a hierarchical ConvGNN for graph embedding. This team built upon a spectral ConvGNN to provide an effective representation learning scheme for end-to-end graph classification tasks. More specifically, they proposed a framework for learning graph feature embeddings while also taking the network architecture and relationships between subjects into account. Deng et al. [61] introduced a multilevel framework to enhance the scalability and accuracy of embedding in an unsupervised manner. The model initially generates a new, efficient graph that contains information about both the node properties and the topology of the original graph. It then repeatedly aggregates the nodes with high spectral similarity, breaking the graph down into numerous smaller graphs.

2) Spatial-based Approaches: Spatial-based methods work on the local neighborhood of nodes, aggregating node representations from neighboring nodes to understand their properties. ConvGNNs of this kind imitate the convolution operations of CNNs by defining convolutions directly in the graph domain. Unlike spectral-based approaches, which are relatively expensive and time-consuming to compute, the structure of the spatial-based approaches is simple and has generated cutting-edge outcomes with graph summarization challenges [73].

As a closely-related approach to Kipf and Welling's model [59], GraphSAGE extends their framework to the inductive setting [64]. GraphSAGE was the first approach to introduce node-wise sampling coupled with minibatch training for node embeddings using spatial convolutions. The updated propagation rule, which uses a mean aggregator function, is formulated as follows [64]:

$$h_i^{(l+1)} = \sigma(W^{(l+1)} \cdot \text{AGGREGATE}(\{h_j^{(l)}, \forall j \in \mathcal{N}_i\})) \quad (20)$$

where a mean aggregator function, AGGREGATE$(\cdot)$, combines the features of node i and its sampled neighbors. The aggregation function can be a mean, max-pooling, or any other form of aggregation. For example, the mean aggregation can be defined as:

$$\text{AGGREGATE}(\{h_j^{(l)}, \forall j \in \mathcal{N}_i\}) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} h_j^{(l)} \quad (21)$$

where $h_j^{(l)}$ is the feature vector of node j at layer l and $|\mathcal{N}_i|$ is the number of sampled neighbors of node i.

The updated feature $h_i^{(l+1)}$ of node i is obtained by applying a learnable weight matrix $W^{(l+1)}$ to the aggregated feature representation:

$$h_i^{(l+1)} = \sigma(W^{(l+1)} \cdot \text{AGGREGATE}(\{h_j^{(l)}, \forall j \in \mathcal{N}_i\})) \quad (22)$$

where $h_i^{(l+1)}$ is the updated feature vector of node i in the $(l+1)$-th layer. $W^{(l+1)}$ is the weight matrix for the $(l+1)$-th layer. And, $\sigma(\cdot)$ is the activation function (e.g., ReLU) applied element-wise. The resulting vector has twice the dimensionality of the input feature vector, as it contains both the original node features and the aggregated neighbor features. GraphSAGE is a more flexible model that allows for different types of aggregator functions to be used. This makes it a good choice for graphs with heterogeneous node types, where different types of information need to be aggregated differently. GraphSAGE has also been shown to perform well on tasks such as graph classification and node classification, both of which are closely tied to the task of graph summarization.

Chen et al. [65] introduced FASTGCN, a node-wise sampling approach that utilizes importance sampling for better summaries. By sampling only a fraction of nodes and leveraging importance weights, FASTGCN approximates the graph convolution operation while maintaining a high level of performance. This results in faster convergence during training, making it particularly suitable for large-scale graph-based tasks. Later, Huang et al. [74] and Zeng et al. [66] proposed layer-wise and graph-wise sampling methods, respectively, to further improve performance. Huang et al. [74] focused on addressing redundancy in node-wise sampling, while Zeng et al.'s GraphSAINT [66] aimed to correct bias and variance in minibatch estimators when sampling subgraphs for training. GraphSAINT is particularly useful for graph-based tasks where dealing with large-scale graphs can be computationally challenging. Its sampling-based approach and minibatch correction mechanism make it a powerful tool for scalable and accurate graph summarization. In a recent variation, Li et al. [75] introduced bipartite GraphSAGE, tailored for bipartite graphs containing different node types and inter-layer edges. This framework involves a user-item graph supporting both user and item embedding, with nodes embedded into two separate feature spaces — one for user-related information and the other for item-related information.

Many aggregation-based methods have also been introduced to summarize graphs without sacrificing too much information within the original graphs. The summarized graph can then be used to assist in further network analysis and graph mining tasks. For instance, Yan et al. [67] introduced

a novel approach called GroupINN, which enhances ConvGNN through summarization, leading to faster training, data denoising, and improved interpretability. They employed an end-to-end neural network model with a specialized node grouping layer, which effectively summarizes the graph by reducing its dimensionality. Hu et al. [76] took structural similarity into consideration, aggregating nodes that share a similar structure into hypernodes. The summarized graph is then refined to restore each node's representation. To this end, a deep hierarchical ConvGNN (H-GCN) architecture with several coarsening operation layers followed by several refinement operation layers performs semi-supervised node classification. The refinement layers are designed to restore the structure of the original graph for graph classification tasks.

The most recent developments in ConvGNNs demonstrate the exciting potential graph summarization holds for a range of applications in healthcare and human motion analysis [68], [77]. Wen et al. [68], for example, presented a promising approach to diagnosing autism spectrum disorder by parsing brain structure information through a multi-view graph convolution network. Dang et al. [77] introduced a new type of graph convolution network, called a multi-scale residual graph convolution network, that shows superior performance in predicting human motion compared to other state-of-the-art models.

C. GAE-based Approaches

An autoencoder is a neural network that consists of an encoder and a decoder. Generally, the encoder transforms the input data into a condensed representation, while the decoder reconstructs the actual input data from the encoder's output [85]. Graph autoencoders, or GAEs, are a type of GNN that can be applied over graph structures, allowing the model to learn a compact and informative representation of a graph. Lately, GAEs have garnered increasing interest for their ability to summarize graphs due to their significant potential for dimensionality reduction [36].

The structure of the encoder and decoder in a GAE can vary depending on the specific implementation and the complexity of the graph data. Generally, both the encoder and decoder are neural network architectures that are designed to process graph data efficiently [86]. The architecture of the encoder may include multiple layers of graph convolutions or aggregations, followed by non-linear activation functions. The output of the encoder is a compact and informative representation of the graph in the latent space. On the other hand, the decoder takes the latent representation obtained from the encoder and reconstructs the original graph structure from it. The decoder's architecture should mirror the encoder's architecture in reverse. It transforms the latent representation back into a graph structure [11].

The goal of a GAE is to learn an encoder and decoder that reduces the gap between the original graph and the reconstruction error of the decoded graph, while also encouraging the latent representation to capture meaningful information about the graph's structure [87]:

$$Z = f_E(A, X), G' = f_D(A, Z) \qquad (23)$$

where G' represents the reconstructed graph, which can consist of either reconstructed features, graph structure, or both. A is the adjacency matrix, and X is the input node feature matrix. f_E serves as the graph encoder, responsible for transforming the graph and node features into a condensed representation. Conversely, f_D acts as the graph decoder, responsible for reconstructing the original graph or its components from the latent representation.

GAEs can be trained using various loss functions, such as mean squared error (MSE) or binary cross-entropy (BCE). They can also be extended to incorporate additional constraints or regularization techniques to improve the quality of the learned graph representation [88]. For example, for graph reconstruction, the goal is to minimize the difference between the original adjacency matrix A and the reconstructed adjacency matrix $\hat{A}$. The MSE loss is calculated as follows [78]:

$$L_{MSE} = \frac{1}{N \times N} \sum_{i,j} (A_{ij} - \hat{A}_{ij})^2 \qquad (24)$$

where, N is the number of nodes in the graph, and A_{ij} and $\hat{A}_{ij}$ are the elements of the original and reconstructed adjacency matrices, respectively.

The majority of GAE-based approaches for graph summarization use combined architectures that include ConvGNNs or RecGNNs [78], [89], [80]. For example, Kipf et al. [78] proposed a variational graph autoencoder (VGAE) for undirected graphs based on their previous work on spectral convolutions [59]. VGAE incorporates a two-layer ConvGNN model based on the variational autoencoder in [89]. The main concept of VGAE is to represent the input graph data not as a single point in the latent space but as a probability distribution. This distribution captures the uncertainty and variability in the graph's latent representation. Instead of directly obtaining a fixed latent representation from the encoder, VGAE samples a random point from the learned distribution. The encoder in VGAE typically consists of two or more graph convolutional layers that process the input graph data and produce latent node representations. Each graph convolutional layer can be defined as follows [78]:

$$Z^{(l+1)} = \sigma(\widetilde{D}^{-1/2} \widetilde{A} \widetilde{D}^{-1/2} Z^{(l)} W^{(l)}) \qquad (25)$$

where $Z^{(l)}$ represents the latent node representations at layer l of the encoder. $\widetilde{A}$ is the adjacency matrix of the graph with added self-loops. $\widetilde{D}$ is the diagonal degree matrix of $\widetilde{A}$. $W^{(l)}$ is the weight matrix for the lth layer. $\sigma(\cdot)$ is the activation function (e.g., ReLU) applied element-wise.

The VGAE introduces stochasticity to GAEs by sampling the latent representation Z from a Gaussian distribution in the latent space. The mean μ and log-variance $\log \sigma^2$ of the distribution are obtained from the output of the last graph convolutional layer:

$$\mu = Z^{(L)} \cdot W^{(\mu)} \qquad (26)$$

$$\log \sigma^2 = Z^{(L)} \cdot W^{(\sigma)} \qquad (27)$$

Here, L represents the last layer of the encoder. $W^{(\mu)}$ and $W^{(\sigma)}$ are learnable weight matrices for obtaining the mean and log-variance, respectively.

TABLE III
COMPARATIVE ANALYSIS OF GAE-BASED APPROACHES FOR GRAPH SUMMARIZATION.

Ref.	Model Name	Evaluation	Performance Metrics	Training Data	Advantages	Limitations
Kipf et al. [78]	VGAE	Link Prediction	AP, AUC	Labeled, Unlabeled	Generative model, unsupervised learning, scalable to large graphs, variational inference.	Limited expressive power, dependency on the quality and quantity of the training data, graph structure assumptions.
Wang et al. [79]	MGAE	Graph Clustering	Accuracy, F1-score, Precision, Recall, ARI	Labeled	Integration of structure and content information, deep marginalized architecture, efficient training procedure.	Performance on recall metric, limited improvement over other algorithms, static setting for structure and content.
Hajiramezanali et al. [80]	VGRNN	Link Prediction	AP, AUC	Labeled, Unlabeled	Flexibility in modeling, interpretable latent representation, incorporation of stochastic latent variables.	Marginal improvement with more flexible priors, challenges in predicting very sparse graphs, tractability of direct optimization.
Fan et al. [81]	One2Multi	Graph Clustering	Accuracy, AUC, NMI, ARI	Labeled, Unlabeled	Effective fusion of multi-view information, joint optimization of embedding and clustering.	Time-consuming, introduction of noise, limited capacity for deep relations.
Cai et al. [82]	GRAE	Graph Clustering	Accuracy, AUC, NMI, ARI	Labeled, Unlabeled	Effective fusion of multiple views, adaptive weight learning, self-training clustering.	Parameter sensitivity, dataset dependency, assumption of homogeneous graphs.
Salha et al. [83]	Linear AE	Link Prediction, Node Clustering	Average Precision, AUC-ROC, AMI	Labeled, Unlabeled	Analysis on 17 real-world graphs with various sizes and characteristics, one-hop interactions.	Performance variation across datasets, relevance of benchmark datasets, limited evaluation of deeper models.
Mrabah et al. [84]	R-GAE	Graph Clustering	Accuracy, NMI, ARI	Labeled	Improved clustering performance, controlled trade-off between FR and FD, theoretical and empirical support, organized approach, and flexibility in integration.	Cannot incorporate structural and content information, trade-off between fr and fd, not suitable for generating graph-specific outputs.

To sample from the Gaussian distribution, the reparameterization trick [90] is used. A random noise vector ϵ is sampled from a standard Gaussian distribution ($\epsilon \sim \mathcal{N}(0, 1)$). The sampled latent representation Z is then computed as:

$$Z = \mu + \epsilon \cdot \exp(\frac{1}{2}\log\sigma^2) \qquad (28)$$

Finally, the decoder maps the sampled latent representation Z back into the graph space. In VGAE, the reconstruction is typically performed using an inner product between the latent node representations to predict the adjacency matrix $\hat{A}$:

$$\hat{A} = \sigma(Z \cdot Z^T) \qquad (29)$$

Here, $\sigma(\cdot)$ is the sigmoid activation function to ensure that the predicted adjacency matrix $\hat{A}$ is within the range [0, 1].

The loss function in VGAE consists of two terms: a reconstruction loss and a kullback-leibler (KL) divergence loss [91]. The reconstruction loss measures the difference between the predicted adjacency matrix $\hat{A}$ and the actual adjacency matrix A. The KL divergence loss penalizes the deviation of the learned latent distribution from the standard Gaussian distribution. The overall loss function is the sum of these two losses as follows:

$$L = \mathbb{E}_{q(Z|X,A)}[logp(A|Z)] - KL[a(Z|X,A)||p(Z)] \qquad (30)$$

As an extension to VGAE, Hajiramezanali et al. [80] constructed a variational graph RNN by integrating a RecGNN and a GAE to model the dynamics of the node attributes and the topological dependencies. The aim of the model is to learn an interpretable latent graph representation as well as to model sparse dynamic graphs.

There are also several aggregation-based approaches built on GAEs. These are generally designed to formulate the challenges with graph clustering tasks as a summarization problem [82], [79], [81], [84]. For example, Cai et al. [82] suggested a graph recurrent autoencoder model for use in clustering attributed multi-view graphs. The fundamental concept behind the approach is to consider both the characteristics that all views have in common and those that make each graph view special. To this end, the framework includes two separate models: the Global Graph Autoencoder (GGAE) and the Partial Graph Autoencoder (PGAE). The purpose of the GGAE is to learn the characteristics shared by all views, while the PGAE captures the distinct features. The cells are grouped into clusters using a soft K-means clustering algorithm after the output is obtained. Fan et al. [81] introduced the One2Multi Graph Autoencoder (OMGAE) for multi-view graph clustering. OMGAE leverages a shared encoder to learn a common representation from multiple views of a graph and uses multiple decoders to reconstruct each view separately. Additionally, OMGAE introduces a new attention mechanism that assigns different weights to each view during the clustering process based on their importance. The model is trained to minimize a joint loss function that considers both the reconstruction error and the clustering performance. Mrabah et al. [84] devised a new graph autoencoder model for attributed graph clustering called GAE-KL. The model uses a new formulation of the objective function, which includes a KL-divergence term, to learn a disentangled representation of the graph structure and the node attributes. The disentangled representation is then used to cluster the nodes based on their similarity in terms of both structure and attributes. The authors also introduced a new evaluation metric called cluster-based classification accuracy (CCA) to measure clustering performance.

Recently, Salha et al. [83] proposed a graph autoencoder architecture that uses one-hop linear models to encode and decode graph information. The approach simplifies the model

while still achieving high performance with graph summarization tasks, such as node clustering and graph classification. Uniquely, this paper presents a direction for designing graph autoencoder models that balances performance with simplicity.

D. GAT-based Approaches

The idea of an attention mechanism was first proposed by Bahdanau and his colleagues in 2014 [98]. The goal was to allow for modelling long-term dependencies in sequential data and to improve the performance of autoencoders. Essentially, attention allows the decoder to concentrate on the most relevant part of the input sequence with the most relevant vectors receiving the highest weights. Graph attention networks or GATs [92] are based on the same idea. They use attention-based neighborhood aggregation, assigning different weights to the nodes in a neighborhood. This type of model is one of the most popular GNN models for node aggregation, largely because it reduces storage complexity along with the number of nodes and edges. The key formulation for a GAT is:

$$h_i = \sigma\left(\sum_{i \in N(j)} \alpha_{ij} W h_j\right) \quad (31)$$

where h_i is the hidden feature vector of node u_i, $N(j)$ is the set of neighbouring nodes of u_i, h_j is the hidden state of neighbouring node u_j, W is a weight matrix, and α_{ij} is the attention coefficient that measures the importance of node u_j to node u_i. The attention coefficients are computed as:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in N_i} exp(e_{ik})} \quad (32)$$

where e_{ij} is a scalar energy value computed as:

$$e_{ij} = LeakyReLU(a^T.[Wh_i || Wh_j]) \quad (33)$$

where a is a learnable parameter vector, and $||$ denotes concatenation. The $LeakyReLU$ function introduces non-linearity into the model and helps prevent vanishing gradients. The softmax function normalizes the energy values across all neighboring nodes so that the attention coefficients sum to one.

By computing attention coefficients for neighboring nodes, GATs are able to selectively focus on the most important parts of the graph for each node. This allows the model to adaptively adjust to different graph structures and tasks. The attention mechanism also means GATs can incorporate node and edge features into the model, making them well-suited to summarization tasks, such as node classification with complex graphs [99].

Today, GATs are considered to be one of the most advanced models for learning with large-scale graphs. However, recently Brody et al. [39] argued that GATs do not actually compute dynamic attention; rather, they only compute a restricted form of static attention. To support their claim, they introduced GATv2, a new version of this type of attention network, which is capable of expressing problems that require computing dynamic attention. Focusing on the importance of dynamic weights, these authors argue that the problem of only supporting static attention can be fixed by changing the sequence of

internal processes in the GAT equations, as shown in Equation 34.

$$e_{ij} = a^T LeakyReLU(W.[h_i || h_j]) \quad (34)$$

As another variation of GAT, Xie et al. [93] proposed a novel multi-view graph attention network named MGAT, to support low-dimensional representation learning based on an attention mechanism in a multi-view manner. The authors focus on a view-based attention approach that not only aggregates view-based node representations but also integrates various types of relationships into multiple views.

Tu et al. [95] explored the benefits of using graph summarization and refining bipartite user-item graphs for recommendation tasks. They applied a conditional attention mechanism to task-based sub-graphs to determine user preferences, which emphasizes the potential of summarizing and enhancing knowledge graphs to support recommender systems. Salehi et al. [94] defined a model based on an autoencoder architecture with a graph attention mechanism that learns low-dimensional representations of graphs. The model compresses the information in the input graph into a fixed-size latent vector, which serves as a summary of the entire graph. Through the use of attention, the model is able to discern and prioritize critical nodes and edges within the graph, making it more effective at capturing the graph's structural and semantic properties.

More recent works on GATs conducted by Chen et al. [96] and Li et al. [97] demonstrate the potential of graph attention networks for summarizing and analyzing complex graph data in various domains. Chen et al. proposed a multi-view graph attention network for travel recommendations. The model takes several different types of user behaviors into account, such as making hotel reservations, booking flights, and leaving restaurant reviews, and, in the process, learns an attention mechanism to weigh the importance of different views for a recommendation. Li et al. developed a multi-relational graph attention network for knowledge graph completion. The model integrates an attention mechanism and edge-type embeddings to capture the complex semantic relations between entities in a knowledge graph.

V. GRAPH REINFORCEMENT LEARNING

Reinforcement learning (RL) is a mathematical model based on sequential decisions that allows an agent to learn via trial and error in an interactive setting through feedback on its actions. Due to the success and fast growth of reinforcement learning in interdisciplinary fields, scholars have recently been inspired to investigate reinforcement learning models for graph-structured data, i.e., graph reinforcement learning or GRL [100]. GRL is largely implemented based on the Bellman theory [101], where the environment is represented as a graph, nodes represent states, edges represent possible transitions between states, and rewards are associated with specific state-action pairs or nodes. The key components of GRL are as follows [100]:

- Environment ($graph$): The graph G represents the environment in which the agent operates. It is defined as $G = (V, E)$, where V is the set of nodes representing

TABLE IV
COMPARATIVE ANALYSIS OF GAT-BASED APPROACHES FOR GRAPH SUMMARIZATION.

Ref.	Model Name	Evaluation	Performance Metrics	Training Data	Advantages	Limitations
Velivckovic et al. [92]	GAT	Node Classification	Accuracy, Micro F1-score	Labeled, Unlabeled	Adaptive attention mechanism for focusing on relevant nodes, the ability to capture long-range dependencies in graphs, scalability to handle large graph structures.	Computational complexity for large graphs, memory-intensive training, susceptible to over-smoothing.
Xie et al. [93]	MGAT	Node Classification, Link Prediction	Micro F1-score, AUC	Labeled	End-to-end multi-view graph embedding framework, attention-based integration of node information, effective and efficient performance, and generalizability to complex graph networks.	Lack of consideration for temporal dynamics, limited scalability to complex graph networks, overfitting risk with large regularization terms, lack of comparison with other multi-view embedding methods.
Salehi et al [94]	GATE	Transductive and Inductive Node Classification	Accuracy	Labeled	Inductive learning capability, a flexible and unified architecture, efficiency and scalability, and comprehensive quantitative and qualitative evaluation.	Dependency on graph structure, lack of label information, and the need for unified architectures for both transductive and inductive tasks.
Brody et al [39]	GAT v2	Node Classification, Link Prediction	Accuracy, ROC-AUC	Labeled, Unlabeled	Addresses the limitations of the GAT model, more robust to noise, can handle more complex interactions between nodes, improved accuracy.	Problem and dataset dependence, difficulty in predicting best architecture, performance gap between theoretical and practical models.
Tu et al [95]	KCAN	Recommender Systems	Hit@k, NDCG@k, AUC	Labeled	Effective knowledge graph distillation, knowledge graph refinement, significant improvements, preserving local preference.	Sampling bias, time complexity, explainability, scalability, hyperparameter sensitivity.
Chen et al [96]	MV-GAN	Recommender Systems	Recall@k, NDCG@k	Labeled, Unlabeled	Leveraging to address data sparsity, multi-view graph embedding, view-level attention mechanism, interpretability.	Limited consideration of factors, complexity and scalability, limited consideration of multiple modalities.
Li et al [97]	MRGAT	Graph Classification, Node Classification, Link Prediction	MR, MRR, Hits@k	Labeled, Unlabeled	Selective aggregation of informative features, effective fusion of entity and relation features, interpretability, and consideration of computational efficiency.	Complexity and redundant computation, sampling useful incorrect training examples, exploiting other background knowledge, influence of attention head.

states and E is the set of edges representing possible transitions between states.

- State (s): In GRL, a state s corresponds to a specific node in the graph. Each node may have associated attributes or features that provide information about the state.
- Action (a): An action a corresponds to a decision or move that the agent can make when in a particular state (node). In graph-based environments, actions can be related to traversing edges between nodes or performing some operation on a node.
- Transition Model (T): The transition model defines the dynamics of the graph, specifying the probability of moving from one state (node) to another by taking a specific action (edge).
  $T(s, a, s') = P(s'|s, a)$, where s is the current state (node), a is the action (edge), and s' is the next state (node).
- Reward Function (R): The reward function defines the immediate reward the agent receives after taking a particular action in a given state (node).
  $R(s, a)$ = Expected immediate reward received when taking action a in state s.
- Policy (π): Similar to standard RL, the policy in Graph RL is a strategy that the agent uses to decide which action to take in each state (node).

$\pi(a|s)$ = Probability of taking action a in state s.
- Value Function (V) and Q-function (Q): In Graph RL, the value function V(s) and the Q-function $Q(s, a)$ represent the expected cumulative reward the agent can obtain starting from a particular state (node) and following a policy π, or by taking action a in state s and then following policy π, respectively. The Q-learning algorithm can be formulated as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (35)$$

where at each timestep t, the state s_t interacts with the environment using a behavior policy based on the Q-table values. It takes action a, receives a reward R, and transitions to a new state s_{t+1} based on the environment's feedback. This process is used to update the Q-table iteratively, continually incorporating information from the new state s_{t+1} until reaching the termination time t.

The primary objective in GRL is to acquire a policy that maximizes the expected Q-function $Q(s, a)$ over a sequence of actions, the target policy is defined as [102]:

$$\pi^* = \arg\max_\pi Q(s, a)$$
$$= \arg\max_\pi \mathbb{E}_{\pi, T}[\sum_{k=0} \gamma^k r_{t+k}|s_t = s, a_t = a] \quad (36)$$

where $\mathbb{E}_{\pi,T}[\cdot]$ denotes the expectation with respect to both the policy π and the distribution of transitions T (i.e., state transitions and rewards). The expression $\sum_{k=0} \gamma^k r_{t+k}$ represents the sum of discounted rewards obtained in the future starting from time step t (the current time step) and continuing for k steps into the future. r_{t+k} is the reward obtained at time step $t+k$ after taking action a_t at time step t and following policy π thereafter. The discount factor γ is a value between 0 and 1 that determines the importance of immediate rewards compared to future rewards. It discounts future rewards to make them less significant than immediate rewards. Smaller γ values make the agent more myopic, whereas larger γ values make the agent more far-sighted. The overall objective is to find the policy π that maximizes the expected sum of rewards (the Q-function) starting from state s and taking action a.

Achieving this goal involves employing various algorithms, like Q-learning, or utilizing a policy gradient method that updates Q-values or policy parameters based on observed rewards and transitions [86].

GRL employs a diverse range of algorithms, and it frequently utilizes GNNs to efficiently process and learn from data structured as graphs. GNNs play a crucial role in updating node representations by considering their neighboring nodes, and they are seamlessly integrated into the RL framework to handle tasks specific to graphs with effectiveness. They are seamlessly integrated into the graph summarization framework to effectively handle tasks that involve summarizing graph structures. For instance, Yan et al. [103] introduced a ConGNN-based neural network specifically designed for graph sampling, enabling the automatic extraction of spatial features from the irregular graph topology of the substrate network. To optimize the learning agent, they adopt a popular parallel policy gradient training method, enhancing efficiency and robustness during training. Wu et al. [104] tackled the problem of graph signal sampling by formulating it as a reinforcement learning task in a discrete action space. They use a deep Q-network (DQN) to enable the agent to learn an effective sampling strategy. To make the training process more adaptable, they modify the steps and episodes. During each episode, the agent learns how to choose a node at each step and selects the best node at the end of the episode. They also redefine the actions and rewards to suit the sampling problem. In another work by Wu et al. [105], a reinforced sample selection approach for GNNs' transfer learning is proposed. The approach uses GRL to guide transfer learning and reduce the divergence between the source and target domains.

There is also a line of GRL research that seeks to use this paradigm to evaluate and improve the quality of graph summaries. For example, Amiri et al. [106] introduced a task-based GRL framework to automatically learn how to generate a summary of a given network. To provide an optimal solution for finding the best task-based summary, the authors made use of CNN layers in combination with a reinforcement learning technique. To improve the quality of the summary, the authors later proposed NetReAct [107], an interactive learning framework for graph summarization. The model uses human feedback in tandem with reinforcement learning to improve the summaries, while visualizing the document network.

In another study, Wickman et al. [108] recently presented a graph sparsification framework, SparRL, empowered by a GRL to be used for any edge sparsification assignment with a specific target for reduction. The model takes an edge reduction ratio as its input, and a learning model decides how best to prune the edges. SparRL proceeds in a sequential manner, removing edges from the graph until a total of edges have been pruned. In another work, Wu et al. [48] introduced GSGAN, a novel method for graph sparsification in community detection tasks. GSGAN excels at identifying crucial relationships not apparent in the original graph and enhances community detection effectiveness by introducing artificial edges. Employing a generative adversarial network (GAN) model, GSGAN generates random walks that effectively capture the network's underlying structure. What sets this approach apart is its utilization of reinforcement learning, which enables the method to optimize learning objectives by deriving rewards from a specially designed reward function. This reinforcement learning component guides the generator to create highly informative random walks, ultimately leading to improved performance in community detection tasks. Yan et al. [111] introduced a GRL approach to summarize geographic knowledge graphs. To obtain a more thorough understanding of the summarizing process, the model exploits components with spatial specificity and includes both the extrinsic and the intrinsic information in the graph. The authors also discuss the effectiveness of spatial-based models and compare the results of their model with models that include non-spatial entities.

Recently, many articles have discussed the potential of using GNN-based GRLs to summarize and analyze complex graph data in domains like neuroscience and computer vision [109], [110]. For example, Zhao et al. [109] suggested a deep reinforcement learning scheme guided by a GNN as a way to analyze brain networks. The model uses a reinforcement learning framework to learn a policy for selecting the most informative nodes in the network and combines that with a GNN to learn the node representations. Also, Goyal et al. [110] presented a GNN-based approach to image classification that relies on reinforcement learning. The model represents images as graphs and learns graph convolutional filters to extract features from the graph representation. They showed that their model outperforms several state-of-the-art methods on benchmark datasets with both image classification and reinforcement learning tasks.

In Table V, we summarize the key features of representative GRL-based approaches for graph summarization. Evaluation methods, performance metrics, training data, advantages, and limitations are compared among different models.

VI. PUBLISHED ALGORITHMS AND DATASETS

In the following section, we will offer a comprehensive overview of the benchmark datasets, evaluation metrics, and open-source implementations. These critical components are extensively examined and elaborated upon in Sections IV and V of the literature survey. By delving into these aspects, we aim to provide a thorough understanding of the landscape covered in the aforementioned sections.

TABLE V
COMPARATIVE ANALYSIS OF SELECTED GRL-BASED APPROACHES FOR GRAPH SUMMARIZATION.

Ref.	Model Name	Evaluation	Performance Metrics	Training Data	Advantages	Limitations
Wu et al. [104]	DQN	Graph Reconstruction	Accuracy	Labeled, Unlabeled	Efficient graph sampling, RL approach, no need for labeled data, potential for automation, improved reconstruction accuracy.	Limitations on performance on big graphs, sampling set size, the assumptions on the training graph.
Amiri et al. [106]	NetGist	Influence Maximization, Community Detection	$\rho_{netgist}$	Labeled, Unlabeled	Automatic flexible approach to generating meaningful graph summaries for a given set of tasks, generalization to unseen Instances.	Computational complexity, task dependency, limited scope of graph optimization problems, lack of comparison with existing methods.
Amiri et al. [107]	NetReAct	Graph Clustering	$\rho_{netreact}$	Labeled, Unlabeled	Incorporating human feedback, meaningful relationships between groups, multi-scale visualization	The simplicity of non-expert feedback, sparsity and inconsistency of human feedback, scalability to larger document datasets, need for further exploration and development.
Wickman et al. [108]	SparRL	PageRank, community structure	Spearman's ρ index, ARI	Labeled, Unlabeled	Task adaptability, learning efficiency and convergence, flexibility, efficiency, and ease of use.	Involves matrix operations and can be computationally intensive for large graphs, sampling-based techniques, limited to static graph settings.
Wu et al. [105]	RSS-GNN	Transfer Learning	AUC-ROC	Labeled, Unlabeled	Efficient and effective sample selection, alleviates divergence between source and target domain graphs.	Non-differentiable sample selection, computational complexity, generalization to new downstream tasks.
Zhao et al. [109]	BNN-GNN	Graph Classification	Average Accuracy, AUC	Labeled, Unlabeled	Improved classification performance, customized aggregation, effective brain network analysis, flexibility in meta-policy application, robustness to different input types.	Performance variation with input types, generalizability, hyperparameter sensitivity, explainability.
Goyal et al. [110]	GNRL	Graph Classification	Topk Accuracy	Labeled, Unlabeled	Improved image representation, interpretability, efficient training, and scalability.	Lack of robustness in coarsened graph representation, training time and gpu utilization, limited improvement over model-free techniques, lack of generalizability to other environments.

A. Datasets

Both synthetic and real-world datasets are used in the development of the field. Synthetic datasets are created by models based on manually-designed rules, while real-world datasets are collected from actual applications and used to evaluate the performance of proposed methods for practical use. The popular real-world datasets are divided into five categories: citation networks, social networks, user-generated networks, bio-informatics networks and image/neuroimages, and knowledge graphs. Table VI presents an overview of the datasets most commonly utilized in the mentioned categories. Additionally, Figure 4 offers a year-wise development trend for each, providing valuable insights into their evolution and availability over time.

B. Evaluation Metrics

GNNs are typically evaluated through tasks like node classification, graph classification, graph clustering, recommendation, and link prediction. To provide an overview of the evaluation criteria used in each study, we categorized each of the articles we reviewed based on the metrics they used to evaluate their methods. These metrics mostly include accuracy, precision, recall, F1-score, and AUC-ROC. Table VII lists the most-used evaluation metrics and their calculation formulas or descriptions.

C. Open Source Implementations

Tables VIII and IX provide a detailed comparison of selected approaches, showcasing the outcomes of replicating established comparable models through two leading GNN-based graph summarization evaluation techniques: node classification and link prediction. The evaluation is conducted on both static datasets such as Cora and Citeseer, as well as dynamic datasets like Enron and Facebook to assess model performance in evolving graph structures. This comparison encompasses model names, platforms, datasets, metrics, hyperparameters, and implementation sources, with all implementations developed using Python 3.x and popular frameworks like PyTorch, PyTorch Geometric, or TensorFlow.

VII. DISCUSSION AND FUTURE DIRECTIONS

This survey has provided a comprehensive examination of GNNs in the context of graph summarization, focusing on key methodologies such as RecGNN, ConvGNN, GAE, GAT, and the emerging field of GRL. Each of these approaches offers unique perspectives and methodologies for capturing the complex relationships inherent in graph data.

RecGNN-based approaches can capture a substantial amount of information during their recursive neighbourhood expansions by using recurrent units to identify the long-term dependence across layers. Further, this process is quite

TABLE VI
PUBLISHED DATASETS.

Category	Dataset	Publications	URL
Citation Networks	Cora	[59], [78], [99], [79], [65], [74], [60], [61], [76], [94], [83], [53], [84], [47]	https://relational.fit.cvut.cz/dataset/CORA
	Citeseer	[59], [78], [99], [79], [74], [61], [76], [94], [83], [53], [84], [47]	https://relational.fit.cvut.cz/dataset/CiteSeer
	PubMed	[59], [78], [99], [74], [61], [76], [94], [83], [53], [84], [47]	https://relational.fit.cvut.cz/dataset/PubMed_Diabetes
	DBLP	[81], [82], [47], [50]	https://dblp.uni-trier.de/xml/
	ACM	[81], [82], [50]	http://www.arnetminer.org/openacademicgraph
Social Networks	Reddit	[64], [65], [74], [60], [61], [66], [62], [83], [56]	https://github.com/redditarchive/reddit
	IMDB	[81], [82]	https://datasets.imdbws.com/
	Karate	[106]	http://networkdata.ics.uci.edu/data/karate/
	Facebook	[106], [80], [108], [48]	http://snap.stanford.edu/data/ego-Facebook.html
	DNC	[49]	https://github.com/alge24/DyGNN/tree/main/Dataset
	UCI	[49]	https://github.com/alge24/DyGNN/tree/main/Dataset
	Twitter	[93], [108]	http://snap.stanford.edu/data/ego-Twitter.html
User-generated Networks	Amazon	[66], [108]	http://snap.stanford.edu/data/amazon-meta.html
	Yelp	[66], [62], [95], [56]	https://www.yelp.com/dataset
	Epinions	[49]	http://snap.stanford.edu/data/soc-Epinions1.html
	Taobao	[75]	https://tianchi.aliyun.com/dataset/649
	MovieLens	[95], [96]	https://grouplens.org/datasets/movielens/
	Last-FM	[95]	https://grouplens.org/datasets/hetrec-2011/
	Eumail	[48]	https://snap.stanford.edu/data/email-EuAll.html
	Enron	[80]	https://snap.stanford.edu/data/email-Enron.html
	POL. BLOGS	[47]	https://networks.skewed.de/net/polblogs
Bio-informatic Networks, Image/Neuroimage	PPI	[64], [99], [61], [66], [62], [83], [56]	https://github.com/williamleif/GraphSAGE
	MUTAG	[45], [46], [94]	https://networkrepository.com/Mutag.php
	PTC	[45]	http://www.predictive-toxicology.org/ptc/
	ENZymes	[45], [46]	https://github.com/snap-stanford/GraphRNN/tree/master/dataset/ENZYMES
	NCI	[45], [46]	https://cdas.cancer.gov/
	Flickr	[66], [62]	https://shannon.cs.illinois.edu/DenotationGraph/
	fMRI	[67], [109]	https://adni.loni.usc.edu/data-samples/access-data/
	ADNI	[63]	https://adni.loni.usc.edu/data-samples/access-data/
	ABIDE	[63]	https://fcon˙1000.projects.nitrc.org/indi/abide/
Knowledge Graphs		[111], [59], [76], [103]	
Synthetic Networks		[106], [107], [40], [39], [56]	

efficient. Notably, RecGNN models can also improve numerical stability during training if they incorporate convolutional filters. However, they may face challenges in long-range dependency modeling due to the vanishing gradient problem, a common issue in recurrent architectures.

ConvGNN-based approaches leverage a more spatial approach, effectively aggregating local neighborhood information. This method has been particularly effective in tasks where local structure is highly informative. Nonetheless, the convolutional approach may not fully capture the global context, which can be critical in certain summarization tasks. In addition, most existing ConvGNN models for graph summarization simply presume the input graphs are static. However, in the real world, dynamically evolving graphs/networks are more common. For instance, in a social network the number of users, their connected friends, and their activities are constantly changing. To this end, learning ConvGNNs on static graphs may not yield the best results. Hence, more research on dynamic ConvGNN models is needed to increase the quality of summaries with large-scale dynamic graphs.

GAE-based approaches offer a powerful framework for unsupervised learning on graphs. By learning to encode and decode graph data, GAEs can generate compact representations that preserve essential topological information. However, the quality of the summarization is heavily dependent on the choice of the encoder and decoder, which can be a non-trivial design choice. In addition, most GAE-based approaches are typically unregularized and mainly focus on minimizing the reconstruction error while ignoring the data distribution of the latent space. However, this might lead to poor graph summarization when working with sparse and noisy real-world graph data. Although there are a few studies on GAE regularization [112], more research is needed in this regard.

GAT-based approaches introduce an attention mechanism that allows for the weighting of nodes' contributions to the representation. This approach can adaptively highlight important features and relationships within the graph. While GATs provide a flexible mechanism that can potentially outperform other methods, they may also require more computational resources and can be prone to overfitting on smaller datasets. Given the recent advancements in this area, we expect to see more research in the future on using GATs to create condensed representations of both static and dynamic graphs.

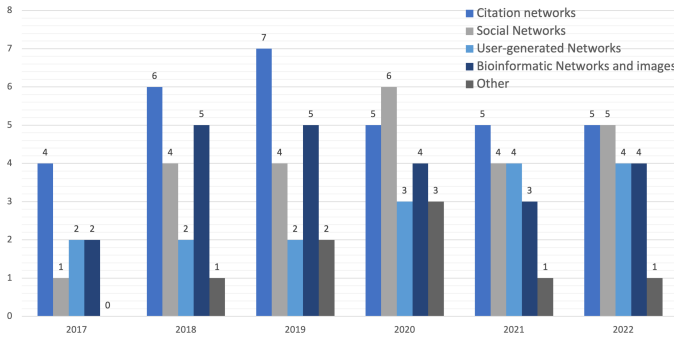**GRL-based** approaches merge reinforcement learning with

Fig. 4. Year-wise development of datasets in the reviewed papers.

TABLE VII
EVALUATION METRICS.

Evaluation Metric	Formula/Description
Accuracy	$\frac{tp+tn}{tp+tn+fp+fn}$
Precision	$\frac{tp}{tp+fp}$
Recall	$\frac{tp}{tp+fn}$
Average Precision	$\sum_{i=1}^{n}(R_i - R_{i-1}) \cdot P_i$
F1-score	$2 \times \frac{Recall \times Precision}{Recall + Precision}$
Micro F1-score	$\frac{tp}{tp+fp+fn}$
Specificity	$\frac{tn}{tn+fp}$
AUC-ROC	The Area Under the ROC Curve.
ARI	Adjusted Rand Index, measures the similarity between two data clusterings.
NMI	The Normalized Mutual Information measures the similarity between two clusters.
Hit@k	$\frac{\text{Number of relevant items in top } k}{k}$
NDCG@k	Normalized Discounted Cumulative Gain at k.
Spearman's ρ index	Measures the strength and direction of the monotonic relationship between two variables.
$\rho_{netgist}$	Expected ratio.
$\rho_{netreact}$	Quantifying the ease of identifying relevant documents.
Mean Rank	$\frac{1}{n}\sum_{i=1}^{n} rank_i$
Mean Reciprocal Rank	$\frac{1}{N}\sum_{i=1}^{N} \frac{1}{rank_i}$

graph models to selectively summarize graphs by learning from reward feedback. This method is promising for decision-centric summarization tasks like graph compression or key substructure identification. Designing customized deep GRL architectures for the purpose of graph summarization stands to be a promising direction in the future. However, being relatively new, GRL faces challenges in defining rewards and efficient exploration of graph spaces.

Across all approaches, we observe a trade-off between the ability to capture different aspects of graph structure and the computational efficiency. Furthermore, each method's performance can vary significantly depending on the characteristics of the dataset in use and the particular summarization task being addressed. Our analysis has revealed that the complexity

of graph data and the early stage of deep learning techniques for graph mining present significant challenges. To address these hurdles and drive future progress, we have identified four potential directions for graph summarization with GNNs.

A. Dynamic Graphs

In the real world, the data that graphs represent can evolve over time, creating changes in a graph's topology, such as new edges that appear, nodes that disappear, or attributes that change over time. These dynamics can cause fundamental changes to the entire graph. Summarizing dynamic graphs typically involves boiling the graph down into a series of snapshots taken at various time increments. The model is then trained over these snapshots, yet a number of challenges can arise. To date, the approaches developed to tackle these problems have primarily focused on capturing temporal patterns and changes in graph topology. However, these methods often struggle with efficiently processing large-scale dynamic graphs and accurately capturing the evolving nature of graph relationships. Future work in this area could focus on developing more scalable algorithms that can handle larger dynamic graphs without compromising processing speed or accuracy.

B. Task-based Summarization

Graph summarization is crucial as graph sizes grow, aiding in understanding, sensemaking, and analysis. Different tasks, like detecting communities or identifying influential nodes, require tailored summarization strategies. This necessitates developing specific approaches for each unique task to effectively identify relevant patterns. Moreover, although task-based summarization is critical, this field of study has seen few successes [106], [107] and is still an underexplored area. Open research problems include how to perform task-based summarization on streaming and heterogeneous graphs and how to leverage human feedback in the learning process.

C. Evaluation Benchmarks

The optimal outcome of a graph summarization process is a "good" summary of the original graph. However, evaluating the "goodness" of a summary is an application-specific task that depends on the task at hand. For example, sampling-based methods are evaluated based on the quality of the sampling, while aggregation-based methods are evaluated based on the quality of classification, and so on. Current studies commonly use comparisons between their method and one or more established methods to measure the quality of their results. For instance, metrics that have been used in the literature include information loss, ease of visualization, and sparsity [2]. However, more and different evaluation metrics are required for cases where the validation process becomes complex and more elements are involved, such as visualization and multi-resolution summaries.

D. Generative Models

In the context of generative models, graph summarization can be used to generate new graphs that have similar properties to the original ones. Generative models, such as VAEs,

TABLE VIII
COMPARABLE MODELS FROM PUBLISHED LITERATURE USING NODE CLASSIFICATION FOR EVALUATIONS. S/D: STATIC/DYNAMIC, NL: NUMBER OF LAYERS, AF: ACTIVATION FUNCTIONS, DR: DROPOUT RATE, LR: LEARNING RATE, WD: WEIGHT DECAY.

	Model	Platform	Dataset	Metrics (%)		Hyperparameters	Repo.
				Accuracy	F1-score		
Node Classification	GCN [59]	PyTorch-Geometric	Cora (S) Citeseer (S)	81.32±0.01 69.30±0.01	81.11±0.01 67.10±0.01	NL, AF, DR, LR, WD, Optimizer	LINK
	GraphSAGE [64]	PyTorch-Geometric	Cora (S) Citeseer (S)	80.20±0.00 69.80±0.02	80.30±0.00 69.70±0.02	NL, AF, DR, LR, WD, Optimizer, Sampling size, Aggregator type	LINK
	GraphSAINT [66]	PyTorch-Geometric	Cora (S) Citeseer (S)	97.16±0.50 91.90±0.40	96.36±0.50 89.66±0.30	NL, AF, DR, LR, WD, Optimizer, Sampling Technique, Sample Size	LINK
	FastGCN [65]	PyTorch-Geometric	Cora (S) Citeseer (S)	79.40±0.05 67.70±0.09	79.45±0.06 66.88±0.12	NL, AF, DR, LR, WD, Optimizer, Layer-wise Importance Sampling	LINK
	GAT [99]	PyTorch-Geometric	Cora (S) Citeseer (S)	81.98±0.00 67.70±0.02	82.00±0.01 67.44±0.02	NL, AF, DR, LR, WD, Optimizer, Attention Mechanism, nHeads, Atten. DR	LINK
	GAT v2 [39]	PyTorch-Geometric	Cora (S) Citeseer (S)	80.90±0.01 67.01±0.03	81.00±0.02 66.30±0.04	NL, AF, DR, LR, WD, Optimizer, Attention Mechanism, nHeads, Atten. DR	LINK
	GATE [94]	Tensorflow	Cora (S) Citeseer (S)	83.10±0.02 71.55±0.02	83.02±0.01 71.88±0.02	NL, AF, LR, Optimizer, Lambda(λ), Weight Sharing	LINK
	H-GCN [76]	Tensorflow	Cora (S) Citeseer (S)	82.44±0.50 71.84±0.60	81.98±0.50 70.96±0.60	NL, AF, LR, Optimizer, Channels, Coarsening Layers, Num. Channel	LINK

TABLE IX
COMPARABLE MODELS FROM PUBLISHED LITERATURE USING LINK PREDICTION FOR EVALUATIONS. S/D: STATIC/DYNAMIC, NL: NUMBER OF LAYERS, AF: ACTIVATION FUNCTIONS, DR: DROPOUT RATE, LR: LEARNING RATE, WD: WEIGHT DECAY.

	Model	Platform	Dataset	Metrics (%)		Hyperparameters	Repo.
				AUC-ROC	AP		
Link Prediction	GAE [78]	PyTorch	Cora (S) Citeseer (S)	89.55±0.02 90.53±0.02	90.98±0.01 91.80±0.02	DR, LR, Optimizer, Encoder, Latent Space Dim., Regularization	LINK
	VGAE [78]	PyTorch	Cora (S) Citeseer (S)	89.27±0.02 91.58±0.03	91.24±0.01 92.50±0.02	DR, LR, Optimizer, Encoder, Regularization, Loss Terms (KL)	LINK
	NetGAN [47]	Tensorflow, Pytorch	Cora (S) Citeseer (S)	81.81±0.00 92.33±0.00	85.42±0.00 92.76±0.00	Regularization, Architectures, Down-Proj, Temp Annealing, RW Length, RW Params	LINK, LINK
	Linear AE [83]	Tensorflow, PyTorch	Cora (S) Citeseer (S)	88.96±1.10 90.88±1.55	89.88±1.00 92.44±1.78	DR, LR, Epochs, Optimizer, Dimensionality d, Hidden Layer Dim.	LINK, LINK
	Linear VAE [83]	Tensorflow, PyTorch	Cora (S) Citeseer (S)	89.77±0.98 91.25±0.90	90.84±0.95 92.66±0.78	DR, LR, Epochs, Optimizer, Dimensionality d, Hidden Layer Dim.	LINK, LINK
	VGRNN [80]	PyTorch	Enron (D) Facebook (D)	92.58±0.25 87.67±0.60	92.66±0.50 87.88±0.70	LR, Epochs, Hidden Layer, GCN Layers, Noise Dimension, Early Stopping	LINK
	SI-VGRNN [80]	PyTorch	Enron (D) Facebook (D)	93.14±0.50 88.04±1.00	93.20±0.50 88.98±1.23	LR, Epochs, Hidden Layer, GCN Layers, Noise Dimension, Early Stopping	LINK

Graph Transformers, Graph Adversarial Networks (GANs), and Graph Auto-Regressive Models, offer effective approaches for graph summarization. These models can learn patterns in graph data and generate new graph summaries by sampling from learned distributions or sequentially generating nodes and edges. Researchers are continually exploring new techniques to pushing the boundaries of model architectures, scalability, controllability, and interpretability [47], [108], [48]. The field presents exciting opportunities for innovation and has the potential to transform various domains through more efficient and accurate graph summarization techniques.

VIII. CONCLUSION

New advancements in deep learning with multi-layer deep neural networks have made it possible to quickly and effectively produce a condensed representation of a large and complex graph. In this paper, we surveyed the technical trends and the most current research in graph summarization with GNNs. We provided an overview of different graph summarization techniques and categorized and described the current GNN-based approaches of graph summarization. We also discussed a new line of research focusing on RL methods to evaluate and improve the quality of graph summaries.

To advance research in this field, we also outlined several frequently-used benchmarking tools, including datasets, open-source codes, and techniques for generating summarized graphs. In addition, we identified four promising directions for future research based on our findings from the survey. We strongly believe that using GNNs for graph summarization is not just a passing trend. Rather, it has a bright future in a wide range of applications across different domains.

As a potential area of focus for our future work, we endeavor to delve into the capabilities of GNN-based generative models including VGAEs [78] and GANs [47], [48], to push the boundaries of graph summarization and generation. Additionally, we will explore the potential of GRL [108] to create new graphs based on their summarized representations. By addressing challenges and expanding the frontiers of graph synthesis, we envision empowering data analysts and researchers with powerful tools for efficient and insightful analysis of complex graph-structured data.

REFERENCES

[1] C. C. Aggarwal and H. Wang, *Managing and mining graph data*. Springer, 2010, vol. 40.

[2] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–34, 2018.

[3] Š. Čebirić, F. Goasdoué, H. Kondylakis, D. Kotzinos, I. Manolescu, G. Troullinou, and M. Zneika, "Summarizing semantic graphs: a survey," *The VLDB Journal*, vol. 28, no. 3, pp. 295–327, 2019.

[4] D. Gibson, R. Kumar, and A. Tomkins, "Discovering large dense subgraphs in massive graphs," in *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005, pp. 721–732.

[5] K. LeFevre and E. Terzi, "Grass: Graph structure summarization," in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 454–465.

[6] P. Zhao, X. Li, D. Xin, and J. Han, "Graph cube: on warehousing and olap multidimensional networks," in *SIGMOD International Conference on Management of Data*, 2011, pp. 853–864.

[7] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 457–464.

[8] B. Karrer and M. E. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, no. 1, p. 016107, 2011.

[9] P. Hu and W. C. Lau, "A survey and taxonomy of graph sampling," *arXiv preprint arXiv:1308.5865*, 2013.

[10] C. Doerr and N. Blenn, "Metric convergence in social network sampling," in *Proceedings of the 5th ACM Workshop on HotPlanet*, 2013, pp. 45–50.

[11] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[12] J. Chen, Y. Saad, and Z. Zhang, "Graph coarsening: from scientific computing to machine learning," *SeMA Journal*, pp. 1–37, 2022.

[13] L.-C. Zhang, "Graph sampling: An introduction," *The Survey Statistician*, vol. 83, pp. 27–37, 2021.

[14] X. Liu, M. Yan, L. Deng, G. Li, X. Ye, and D. Fan, "Sampling methods for efficient training of graph convolutional networks: A survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 2, pp. 205–234, 2021.

[15] Ü. Çatalyürek, K. Devine, M. Faraj, L. Gottesbüren, T. Heuer, H. Meyerhenke, P. Sanders, S. Schlag, C. Schulz, D. Seemaier *et al.*, "More recent advances in (hyper) graph partitioning," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[16] S. A. Bhavsar, V. H. Patil, and A. H. Patil, "Graph partitioning and visualization in graph mining: a survey," *Multimedia Tools and Applications*, vol. 81, no. 30, pp. 43 315–43 356, 2022.

[17] L. Yue, X. Jun, Z. Sihang, W. Siwei, G. Xifeng, Y. Xihong, L. Ke, T. Wenxuan, L. X. Wang *et al.*, "A survey of deep graph clustering: Taxonomy, challenge, and application," *arXiv preprint arXiv:2211.12875*, 2022.

[18] C. Lee and D. J. Wilkinson, "A review of stochastic block models and extensions for graph clustering," *Applied Network Science*, vol. 4, no. 1, pp. 1–50, 2019.

[19] Y. Xie, B. Yu, S. Lv, C. Zhang, G. Wang, and M. Gong, "A survey on heterogeneous network representation learning," *Pattern recognition*, vol. 116, p. 107936, 2021.

[20] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart, "Representation learning for dynamic graphs: A survey," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 2648–2720, 2020.

[21] M. Xu, "Understanding graph embedding methods and their applications," *SIAM Review*, vol. 63, no. 4, pp. 825–853, 2021.

[22] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and S. Y. Philip, "A survey on heterogeneous graph embedding: methods, techniques, applications and sources," *IEEE Transactions on Big Data*, 2022.

[23] W. Jin, L. Zhao, S. Zhang, Y. Liu, J. Tang, and N. Shah, "Graph condensation for graph neural networks," *arXiv preprint arXiv:2110.07580*, 2021.

[24] B. Mayer and B. Perozzi, "Scaling heterogeneous graph sampling and gnns with google cloud dataflow," https://cloud.google.com/blog/products/ai-machine-learning/scaling-heterogeneous-graph-sampling-gnns-google-cloud-dataflow, 2022, accessed: April 4, 2023.

[25] R. Interdonato, M. Magnani, D. Perna, A. Tagarelli, and D. Vega, "Multilayer network simplification: approaches, models and methods," *Computer Science Review*, vol. 36, p. 100246, 2020.

[26] P. S. Chodrow, N. Veldt, and A. R. Benson, "Generative hypergraph clustering: From blockmodels to modularity," *Science Advances*, vol. 7, no. 28, p. eabh1303, 2021.

[27] M. P. Boobalan, D. Lopez, and X. Z. Gao, "Graph clustering using k-neighbourhood attribute structural similarity," *Applied soft computing*, vol. 47, pp. 216–223, 2016.

[28] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[29] M. Kurant, M. Gjoka, C. T. Butts, and A. Markopoulou, "Walking on a graph with a magnifying glass: stratified sampling via weighted random walks," in *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, 2011, pp. 281–292.

[30] A. D. Stivala, J. H. Koskinen, D. A. Rolls, P. Wang, and G. L. Robins, "Snowball sampling for estimating exponential random graph models for large networks," *Social Networks*, vol. 47, pp. 167–188, 2016.

[31] M. Hajiabadi, "Efficient graph summarization of large networks," Ph.D. dissertation, University of Victoria, 2022.

[32] K. A. Kumar and P. Efstathopoulos, "Utility-driven graph summarization," *Proceedings of the VLDB Endowment*, vol. 12, no. 4, pp. 335–347, 2018.

[33] S. Dumbrava, A. Bonifati, A. N. R. Diaz, and R. Vuillemot, "Approximate querying on property graphs," in *Scalable Uncertainty Management: 13th International Conference, SUM 2019, Compiègne, France, December 16–18, 2019, Proceedings 13*. Springer, 2019, pp. 250–265.

[34] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[35] G. Dong, M. Tang, Z. Wang, J. Gao, S. Guo, L. Cai, R. Gutierrez, B. Campbel, L. E. Barnes, and M. Boukhechba, "Graph neural networks in iot: a survey," *ACM Transactions on Sensor Networks*, vol. 19, no. 2, pp. 1–50, 2023.

[36] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[37] F. Liu, S. Xue, J. Wu, C. Zhou, W. Hu, C. Paris, S. Nepal, J. Yang, and P. S. Yu, "Deep learning for community detection: progress, challenges and opportunities," *arXiv preprint arXiv:2005.08225*, 2020.

[38] L. Wu, P. Cui, J. Pei, L. Zhao, and L. Song, "Graph neural networks," in *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer, 2022, pp. 27–37.

[39] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" *arXiv preprint arXiv:2105.14491*, 2021.

[40] P. Goyal, S. R. Chhetri, and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowledge-Based Systems*, vol. 187, p. 104816, 2020.

[41] B. Huang and K. M. Carley, "Inductive graph representation learning with recurrent graph neural networks," *CoRR, abs/1904.08035*, 2019.

[42] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[43] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[44] X. Lai, P. Yang, K. Wang, Q. Yang, and D. Yu, "Mgrnn: Structure generation of molecules based on graph recurrent neural networks," *Molecular Informatics*, vol. 40, no. 10, p. 2100091, 2021.

[45] A. Taheri, K. Gimpel, and T. Berger-Wolf, "Learning graph representations with recurrent neural network autoencoders," *KDD Deep Learning Day*, 2018.

[46] Y. Jin and J. F. JaJa, "Learning graph-level representations with recurrent neural networks," *arXiv preprint arXiv:1805.07683*, 2018.

[47] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "Netgan: Generating graphs via random walks," in *International conference on machine learning*. PMLR, 2018, pp. 610–619.

[48] H.-Y. Wu and Y.-L. Chen, "Graph sparsification with generative adversarial network," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1328–1333.

[49] Y. Ma, Z. Guo, Z. Ren, J. Tang, and D. Yin, "Streaming graph neural networks," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 719–728.

[50] S. Khoshraftar, S. Mahdavi, A. An, Y. Hu, and J. Liu, "Dynamic graph embedding via lstm history tracking," in *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2019, pp. 119–127.

[51] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[52] A. Taheri, K. Gimpel, and T. Berger-Wolf, "Learning to represent the evolution of dynamic graphs with recurrent models," in *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 301–307.

[53] K. Ge, J.-Q. Zhao, and Y.-Y. Zhao, "Gr-gnn: Gated recursion-based graph neural network algorithm," *Mathematics*, vol. 10, no. 7, p. 1171, 2022.

[54] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, pp. 64–67, 2001.

[55] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, "Semantic object parsing with graph lstm," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 125–143.

[56] C.-Y. Zhang, Z.-L. Yao, H.-Y. Yao, F. Huang, and C. P. Chen, "Dynamic representation learning via recurrent graph neural networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 3, pp. 468–479, 2022.

[57] S. Li, K. W. Wong, C. C. Fung, and D. Zhu, "Improving question answering over knowledge graphs using graph summarization," in *International Conference on Neural Information Processing*. Springer, 2021, pp. 489–500.

[58] K. Zarzycki and M. Ławryńczuk, "Advanced predictive control for gru and lstm networks," *Information Sciences*, vol. 616, pp. 229–254, 2022.

[59] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[60] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6861–6871.

[61] C. Deng, Z. Zhao, Y. Wang, Z. Zhang, and Z. Feng, "Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding," *arXiv preprint arXiv:1910.02370*, 2019.

[62] E. Rossi, F. Frasca, B. Chamberlain, D. Eynard, M. Bronstein, and F. Monti, "Sign: Scalable inception graph neural networks," *arXiv preprint arXiv:2004.11198*, vol. 7, p. 15, 2020.

[63] H. Jiang, P. Cao, M. Xu, J. Yang, and O. Zaiane, "Hi-gcn: a hierarchical graph convolution network for graph embedding learning of brain network and brain disorders prediction," *Computers in Biology and Medicine*, vol. 127, p. 104096, 2020.

[64] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*. MIT Press, 2017, pp. 1024–1034.

[65] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," *arXiv preprint arXiv:1801.10247*, 2018.

[66] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *arXiv preprint arXiv:1907.04931*, 2019.

[67] Y. Yan, J. Zhu, M. Duda, E. Solarz, C. Sripada, and D. Koutra, "Groupinn: Grouping-based interpretable neural network for classification of limited, noisy brain data," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 772–782.

[68] G. Wen, P. Cao, H. Bao, W. Yang, T. Zheng, and O. Zaiane, "Mvs-gcn: A prior brain structure learning-guided multi-view graph convolution network for autism spectrum disorder diagnosis," *Computers in Biology and Medicine*, vol. 142, p. 105239, 2022.

[69] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: a comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.

[70] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[71] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[72] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[73] A. Ajit, K. Acharya, and A. Samanta, "A review of convolutional neural networks," in *International Conference on Emerging Trends in Information Technology and Engineering*. IEEE, 2020, pp. 1–5.

[74] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[75] Z. Li, X. Shen, Y. Jiao, X. Pan, P. Zou, X. Meng, C. Yao, and J. Bu, "Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications," in *IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 1677–1688.

[76] F. Hu, Y. Zhu, S. Wu, L. Wang, and T. Tan, "Hierarchical graph convolutional networks for semi-supervised node classification," *arXiv preprint arXiv:1902.06667*, 2019.

[77] L. Dang, Y. Nie, C. Long, Q. Zhang, and G. Li, "Msr-gcn: Multi-scale residual graph convolution networks for human motion prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 11 467–11 476.

[78] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[79] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "Mgae: Marginalized graph autoencoder for graph clustering," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 889–898.

[80] E. Hajiramezanali, A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[81] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2multi graph autoencoder for multi-view graph clustering," in *proceedings of the web conference 2020*, 2020, pp. 3070–3076.

[82] E. Cai, J. Huang, B. Huang, S. Xu, and J. Zhu, "Grae: Graph recurrent autoencoder for multi-view graph clustering," in *2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, 2021, pp. 1–9.

[83] G. Salha, R. Hennequin, and M. Vazirgiannis, "Simple and effective graph autoencoders with one-hop linear models," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*. Springer, 2021, pp. 319–334.

[84] N. Mrabah, M. Bouguessa, M. F. Touati, and R. Ksantini, "Rethinking graph auto-encoder models for attributed graph clustering," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[85] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," in *Machine Learning*. Academic Press, 2020, pp. 193–208.

[86] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2020.

[87] Z. Hou, X. Liu, Y. Dong, C. Wang, J. Tang *et al.*, "Graphmae: Self-supervised masked graph autoencoders," *arXiv preprint arXiv:2205.10803*, 2022.

[88] G. Salha, R. Hennequin, and M. Vazirgiannis, "Keep it simple: Graph autoencoders without graph convolutional networks," *arXiv preprint arXiv:1910.00942*, 2019.

[89] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[90] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," *Advances in neural information processing systems*, vol. 28, 2015.

[91] T. Kim, J. Oh, N. Y. Kim, S. Cho, and S.-Y. Yun, "Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation," in *30th International Joint Conference on Artificial Intelligence (IJCAI-21)*. IJCAI, 2021, pp. 2628–2635.

[92] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[93] Y. Xie, Y. Zhang, M. Gong, Z. Tang, and C. Han, "Mgat: Multi-view graph attention networks," *Neural Networks*, vol. 132, pp. 180–189, 2020.

[94] A. Salehi and H. Davulcu, "Graph attention auto-encoders," in *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2020, pp. 989–996.

[95] K. Tu, P. Cui, D. Wang, Z. Zhang, J. Zhou, Y. Qi, and W. Zhu, "Conditional graph attention networks for distilling and refining knowledge graphs in recommendation," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1834–1843.

[96] L. Chen, J. Cao, Y. Wang, W. Liang, and G. Zhu, "Multi-view graph attention network for travel recommendation," *Expert Systems with Applications*, vol. 191, p. 116234, 2022.

[97] Z. Li, Y. Zhao, Y. Zhang, and Z. Zhang, "Multi-relational graph attention networks for knowledge graph completion," *Knowledge-Based Systems*, vol. 251, p. 109262, 2022.

[98] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[99] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

[100] N. Mingshuo, C. Dongming, and W. Dongqi, "Reinforcement learning on graph: A survey," *arXiv preprint arXiv:2204.06127*, 2022.

[101] R. Bellman and R. E. Kalaba, *Selected papers on mathematical trends in control theory*. Dover Publications, 1964.

[102] M. Nie, D. Chen, and D. Wang, "Reinforcement learning on graphs: A survey," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2023.

[103] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 6, pp. 1040–1057, 2020.

[104] M. Wu, Q. Zhang, Y. Gao, and N. Li, "Graph signal sampling with deep q-learning," in *2020 International Conference on Computer Information and Big Data Applications (CIBDA)*, 2020, pp. 450–453.

[105] B. Wu, X. Liang, X. Zheng, J. Wang, and X. Zhou, "Reinforced sample selection for graph neural networks transfer learning," in *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 2022, pp. 1281–1288.

[106] S. E. Amiri, B. Adhikari, A. Bharadwaj, and B. A. Prakash, "Netgist: Learning to generate task-based network summaries," in *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 857–862.

[107] S. E. Amiri, B. Adhikari, J. Wenskovitch, A. Rodriguez, M. Dowling, C. North, and B. A. Prakash, "Netreact: Interactive learning for network summarization," *arXiv preprint arXiv:2012.11821*, 2020.

[108] R. Wickman, X. Zhang, and W. Li, "Sparrl: Graph sparsification via deep reinforcement learning," *arXiv preprint arXiv:2112.01565*, 2021.

[109] X. Zhao, J. Wu, H. Peng, A. Beheshti, J. J. Monaghan, D. McAlpine, H. Hernandez-Perez, M. Dras, Q. Dai, Y. Li *et al.*, "Deep reinforcement learning guided graph neural networks for brain network analysis," *Neural Networks*, vol. 154, pp. 56–67, 2022.

[110] N. Goyal and D. Steiner, "Graph neural networks for image classification and reinforcement learning using graph representations," *arXiv preprint arXiv:2203.03457*, 2022.

[111] B. Yan, K. Janowicz, G. Mai, and R. Zhu, "A spatially explicit reinforcement learning model for geographic knowledge graph summarization," *Transactions in GIS*, vol. 23, no. 3, pp. 620–640, 2019.

[112] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," *arXiv preprint arXiv:1802.04407*, 2018.

Nasrin Shabani received a Master of Research degree in Computer Science with First Class Honours from the Macquarie University, Sydney, NSW, Australia. She is currently pursuing a Ph.D. in Computer Science at the same institution. Her research interests lie at the intersection of graph mining, graph summarization, and deep learning. Through her work, she aims to develop novel algorithms and techniques that can extract meaningful insights and patterns from complex graph data structures.

Jia Wu (Senior Member, IEEE) is currently the Research Director of the Centre for Applied Artificial Intelligence and the Director of Higher Degree Research in the School of Computing at Macquarie University, Sydney, Australia. Dr Wu received his Ph.D. degree in computer science from the University of Technology Sydney, Australia. His current research interests include data mining and machine learning. Since 2009, he has published 100+ refereed journal and conference papers, including TKDE, TKDD, KDD, ICDM, WWW, and NeurIPS.

Amin Beheshti holds B.S. (1st Hons.) and M.S. degrees (1st Hons.) in computer science and engineering, and a Ph.D. in computer science from UNSW Sydney, Australia. Amin is a Full Professor of data science at Macquarie University. He is currently the Director of the Centre for Applied Artificial Intelligence and the Head of the Data Science Research Laboratory, School of Computing, Macquarie University. He is a leading author of several authored books in data, social, and process analytics, co-authored with other high-profile researchers.

Quan Z. Sheng received his Ph.D. degree in computer science from the University of New South Wales, Sydney, NSW, Australia. He is a Distinguished Professor and Head of the School of Computing, at Macquarie University, Sydney. His research interests include big data analytics, service-oriented computing, and the Internet of Things. Microsoft Academic ranked Prof. Michael Sheng as one of the Most Impactful Authors in Services Computing (ranked Top 5 All-Time) and in the Web of Things (ranked Top 20 All-Time).

Jin Foo is the Staff Data Scientist at Prospa, Australia's top online lender to small businesses and was previously Data Science Lead at Woolworths Group; specialising in identity resolution, hyper-personalised offers, propensity modelling, sequential bin packing and time-series analysis. Jin is currently a 2nd year Master of Research student at the School of Computing, Macquarie University, Sydney, Australia. His research focuses on anomaly detection with word embeddings, hashing algorithms and graph networks.

Venus Haghighi is currently a Ph.D. student in computer science at the School of Computing, Macquarie University, Sydney, NSW, Australia. The focus of her research is to explore classic GNN models and explore robust graph learning paradigms to detect and mitigate the camouflage behavior of malicious actors in both static and dynamic networks. Her research interests include graph-based anomaly detection, graph neural networks, graph-based fraud detection, and graph data mining.

Ambreen Hanif is a 2nd year Ph.D. student at the School of Computing, Macquarie University NSW, Australia. After completing her Master's degree, she decided to continue her research as a Ph.D. candidate. Her research interests lie in the field of Explainable Artificial Intelligence (XAI) for Deep Neural Networks, Data Provenance, and Storytelling with XAI. Specifically, she aims to develop novel methods to enhance the interpretability and transparency of deep neural networks.

Maryam Shahabikargar received her Master of Research degree in computer science from the Macquarie University, Sydney, Australia. After completing her Master's degree, she decided to continue her research as a Ph.D. candidate in computer science at Macquarie University. Her current research interests include not only NLP and graph embeddings but also link prediction and anomaly detection. She aims to develop a model that combines her research interests to tackle a specific problem in the field of finance.