mathematics

MDPI

Article

Enhancing Graph Summarization Using Node Importance and Graph Attention Networks

Krista Rizman Žalik *, Domen Mongus and Mitja Žalik

Faculty of Electrical Engineering and Computer Science, University of Maribor, Koroška cesta 46, 2000 Maribor, Slovenia

* Correspondence: krista.zalik@um.si

Abstract

As the scale of graph-structured data continues to grow, graph summarization has become an important technique for storage efficiency and high-level visualization. This study investigates a Node Importance (NI) approach to graph summarization that prioritizes structural integrity over simple size reduction. The NI approach selects super nodes by ranking vertices through centrality and propagation metrics. Experimental results demonstrate that the proposed NI method achieves compression rates comparable to or slightly lower than traditional Minimum Description Length (MDL) methods across various datasets while maintaining structural integrity. However, today, the high dimensionality and complexity of modern graph data are making deep learning techniques more popular. Great progress in deep learning summarization techniques is achieved with Graph Neural Networks (GNNs). This study investigates the structure and suitability of different GNN architectures for graph summarization using the NI approach. Graph Attention Networks (GATs) and their variants are discussed as a flexible, learned notion of node importance via attention. We present an examination of GATs, covering both diverse approaches and improvements. This study also discusses extensions that enhance the concept of node importance established by the GAT model, GAT variants for node importance estimation, and application-specific GAT research.

Keywords: graph summarization; node importance; graph neural networks; graph attention networks

MSC: 68T07

Academic Editors: Chao Tong and Xiaojun Wang

Received: 2 March 2026
Revised: 5 April 2026
Accepted: 10 April 2026
Published: 12 April 2026

1. Introduction

Large-scale complex systems such as the World Wide Web, the internet, and online social platforms contain huge network data, which are commonly modeled as massive graphs whose size and structural complexity continue to increase rapidly [1]. These networks have high dimensionality, heterogeneous node and edge relationships, and dynamic evolution over time, making traditional graph analytics computationally expensive and unscalable as data volume increases [2]. As the volume of graphs increases, compact and informative summary graph representations become essential for enabling efficient storage, transfer, and processing of graphs.

Graph summarization is an efficient paradigm for addressing these scalability challenges by producing condensed representations that preserve structural or semantic properties of the original graph structures relevant for different tasks such as queries, pattern

mining, or visualization [3]. The aim of graph summarization is to create a compact, smaller representation of a large, complex graph while preserving its essential properties, structural patterns, or query-answering capabilities. Efficient graph summarization methods for condensing and simplifying graph data are thus becoming vital for efficient applications, because executing traditional graph algorithms—such as centrality and shortest path—on a condensed summary graph significantly reduces processing time and increases the performance of applications.

Many methods have been proposed for graph summarization, and there are several recognized challenges in graph summarization [4]. Defining a universally appropriate measure of a good summary is difficult. It is highly application-dependent and often involves a trade-off between the compression ratio and the reconstruction error. Summarizing more complex structures, such as attributed graphs and dynamic or multi-relation graphs, remains challenging in terms of interpretability. Summaries, especially those created by deep learning models, must be interpretable to provide meaningful insights to human analysts.

This study investigates a node importance (NI) approach for graph summarization, shifting the focus from mere size reduction to the preservation of structurally significant entities. The NI approach selects super nodes by ranking vertices through centrality and propagation metrics. We propose an NI method and evaluate its performance. Experimental results demonstrate that the proposed NI method achieves compression rates comparable to or slightly lower than traditional Minimum Description Length (MDL) methods across various. Datasets while maintaining structural integrity.

Deep learning (DL) has become a dominant approach in graph summarization because traditional methods, like MDL or basic clustering, often encounter challenges with the scale and high dimensionality of modern network data, like huge social networks. Graph Neural Networks (GNNs) have become essential for extracting the full potential of graph-structured data. Over the past decade, a wide array of GNN subcategories has emerged to tackle the inherent complexities of graph learning. This study investigates the structure and suitability of different GNN architectures for graph summarization using the NI approach. Graph Attention Networks (GATs) and their variants provide a flexible, learned notion of node importance via attention. This study discusses extensions that enhance the concept of node importance established by the GAT model, GAT variants for node importance estimation, and application-specific GAT research.

We introduce the background and definitions in Section 2. Then, we discuss why node importance generally lowers compression and compare the node importance method with some Minimum Description Length (MDL) methods in Section 3. We detail the comparison with GNNs and GATs in Section 4. Section 5 discusses GAT models. Section 6 discusses extensions that enhance the concept of node importance established by the GAT model, and Section 7 describes GAT variants for node importance estimation. Section 8 discusses application-specific GAT research. We conclude the paper in Section 9.

2. Background and Definitions

In this section, we define the graph summarization problem.

2.1. Complex Systems and Graphs

Large-scale complex systems such as the World Wide Web, the internet, and online social platforms contain huge network data, which are commonly modeled as massive graphs. Most complex networks share scale-free and small-world properties [5].

Small-world property means that any node can be reached from any other in a small number of steps. Complex networks also have high clustering compared with random

graphs. In these networks, if node u is connected to v and v links to z, there is a high probability that u and z are also directly connected. This creates communities.

Scale-free property means that the degree distribution follows a power law. Probability $P(k)$, denoting the degree distribution that describes a node with k links, often follows a power law. It can be represented by a power law with a degree exponent γ usually in the range $2 < \gamma < 3$ (Equation (1)) [6].

$$P(k) = k^{-\gamma} \tag{1}$$

This means most nodes have very few connections, while a few hubs have a lot of connections. Scale-free structure has been reported in the WWW, Internet, metabolic, protein–interaction, and some social networks [7].

Large-scale complex systems contain huge network data, which are commonly modeled as graphs. The graph $G(V, E)$ consists of a set V of n nodes $V = \{v_1, v_2, \ldots v_n\}$ or vertices and a set E (Equation (2)) of m edges or links connecting node pairs. e_{ij} denotes an edge between nodes v_i, v_j.

$$E = \left\{e_{ij}\right\}_{i,j=1}^{n}. \tag{2}$$

Example graph, in Figure 1 has five nodes and five edges.

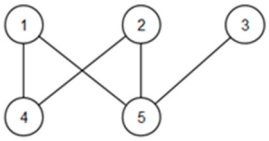


Figure 1. Example graph has five nodes and five edges.

The graph is represented by an adjacency matrix A dimension $n \times n$, where n is the number of nodes, with elements with values being 1 if the edge between nodes denoted by column and row exists in the graph, and 0 otherwise (Equation (3)).

$$A = \left[a_{ij}\right] \; i, j = 1 \ldots n; \; a_{ij} = 1 \; if \; e_{ij} \in E; \quad a_{ij} = 0 \; if \; e_{ij} \notin E; \; i, j = 1 \ldots n \tag{3}$$

In a directed graph, the direction of edges denoted by an arrow describes one edge, while in an undirected graph, an edge represents both edges from the beginning to the end node and from the end to the beginning node of the edge (Equation (4)).

$$a_{ij} = a_{ji} \; \Rightarrow \; directed \; G; \; a_{ij} = a_{ji} \; \Rightarrow \; undirected \; G \tag{4}$$

When edges are associated with weights from set W, the graph is called a weighted graph; otherwise, it is an unweighted graph (Equation (5)).

$$a_{ij} \in \left\{0, w_{ij}\right\}; \quad w_{ij} \in W \; \Rightarrow \; weighted \; G; \quad a_{ij} \in \{0, 1\} \; \Rightarrow \; unweighted \; G \tag{5}$$

G is considered labeled if every edge has an associated label from the set of labels L or if the nodes are also labeled; otherwise, G is unlabeled (Equation (6)).

$$lab(e_{ij}) = l_k \; ; \; lab(v_i) = l_o; \quad e_{ij} \in E \; ; v_i \in V; l_k, \; l_o \; \in L \; \Rightarrow \; labeled \; G \tag{6}$$

2.2. Graph Summarization

The continuous growth of massive, interconnected datasets makes direct analysis impractical, which is solved by the graph summarization with the following benefits [8].

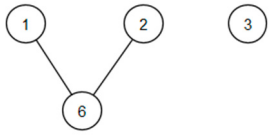
- Graph summarization decreases the disk space required to store the graph.

- Performance is increased by executing traditional graph algorithms—such as centrality and shortest path—on a condensed summary graph to significantly reduce processing time.
- More intuitive visualization and human interpretation are allowed because graph summarization provides greater clarity by simplifying the graph structure.
- Exploratory analysis is simplified by graph summarization that can quickly reveal key structural patterns, communities, and outliers.

Graph summarization has been playing an important role in areas such as network analysis, data mining, machine learning, and visualization.

Given an undirected graph G (V, E) with a set of vertices V and a set of edges E, a summary $G_s = (V_s, E_s)$, with a set of super nodes V_s and super edges E_s, is a condensed representation of the graph. It preserves its key properties and has a smaller number of edges and vertices. $E_s \ll E$; $V_s < V$. We seek a representation $R = (G_s, C)$ that consists of a summary graph and the set of edge corrections C enabling complete graph reconstruction by lossless summarization methods. The summary graph is built on a set of super nodes where each super node is a partition (i.e., disjoint subsets) over a set of input nodes. Each super node represents a set of nodes, and each super edge between two super nodes means all pairs of nodes in the Cartesian product of these two super nodes are linked. The set of edge corrections stores the edges to insert (annotated as '+e') and remove (annotated as '$-e$') when recreating the input graph from the summary graph. Multiple sets of vertices with similar neighbors are grouped in super nodes, while the edge corrections enable a lossless recreation of the input graph from the summary graph. Two summary graphs of the example graph from Figure 1 are shown in Figure 2.

(**a**) Super node: 6(4, 5); correction edges: +(3, 5).



(**b**) Super node: 6(4, 5); correction edges: −(3, 4).

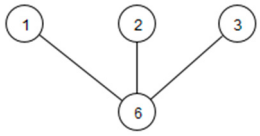


Figure 2. Two different summary graphs (**a**,**b**) of example graph from Figure 1 with the same super edge and different correction edge and edges between nodes.

2.3. Graph Summarization Techniques

Graph summarization, performing graph reduction, can be divided into three main techniques: aggregation, selection, or transformation [9].

Graph coarsening and aggregation represent the most common approach to graph reduction, where nodes are not simply removed but are replaced by new entities. This method identifies groups of highly connected nodes and merges them into a single super node, while their connecting edges are consolidated into super edges. The mechanisms used include node partitioning and the Minimum Description Length (MDL) principle [10]. MDL is a powerful concept from information theory that formalizes the idea that the best explanation for a set of data is the one that provides the shortest description of that data. The result is a smaller, more manageable summary graph composed of super nodes and super edges.

Graph sparsification and selection methods remove objects from the graph without replacing them with new structures. Graph sparsification includes sampling techniques that select specific subsets of nodes and edges from the input, and simplification techniques that remove less important edges. These methods aim to preserve essential graph properties, such as distances, cuts, or the spectrum of the Laplacian matrix.

Graph condensation and transformation convert graph objects into different formats like embedding vectors. Graph condensation includes graph projection, which simplifies multi-layers into single-layer summaries, and graph embedding, which maps the graph into a lower-dimensional space. Graph condensation and transformation are done by deep learning architecture like Graph Neural Networks (GNNs) [11]. The outcome is a smaller, synthetic graph or a numerical representation optimized for machine learning tasks.

3. Node Importance and Graph Summarization

3.1. Node Importance (NI) and Graph Summarization

Below, we summarize how NI-based strategies influence lossless graph compression and why they usually lead to lower compression rates.

NI-based strategies influence lossless graph compression by prioritizing vertices that are structurally, semantically, or operationally significant rather than those that maximize redundancy with the aim of keeping the most important parts of the graph. A problem is how to measure node importance. Node importance in graph summarization methods defines what is preserved and what can be safely merged or removed. There is no universally accepted importance definition, and different tasks, such as querying or visualization, need different notions. Many methods use centrality measures (degree, betweenness, PageRank, eigenvector) to estimate node importance in the graph [12–14].

Centrality-based strategies, such as betweenness, closeness, or PageRank, rank vertices according to their global influence. Betweenness centrality assigns importance $I(v)$ of vertex v: $I(v) = \sum_{(s \neq v \neq t)} \sigma_{st}(v)/\sigma_{st}$ where σ_{st} is the number of shortest paths between vertices s and t, and $\sigma_{st}(v)$ is the number of those paths passing through vertex v. Vertices with high centrality often connect distant parts of the graph, and their encoding scatters edges across and does not consider locality, which reduces the compression rate [13,14].

In degree-based ordering, each vertex v is ranked by its degree $I(v) = deg(v)$, where the degree counts the number of neighbor vertices. High-degree hubs are placed early in the processing to expose shared neighbors. However, these hubs typically have long and diverse adjacency lists, which increases entropy and limits the effectiveness of summarization, resulting in lower compression ratios. A degree-based approach is commonly used in web, citation, and social graphs [12]. To evaluate how a degree centrality-based NI approach impacts graph summarization and compression efficiency, we conducted an experiment described in the next Section.

3.2. Experiment: Node Importance (NI) Lowers Compression

3.2.1. Methods

To evaluate the extent to which the NI approach to graph summarization reduces compression efficiency, we created a graph summarization method based on degree node importance, named ImZip, with the following steps:

Repeat until there are no vertices not assigned to super-vertices in the graph.

- Find the local maximal degree vertex maxVertex (with the greatest number of neighbors).
- Add neighbor vertices with a degree greater than half of the degree of maxVertex and form core vertices.
- Edges between core vertices are removed, and edges that do not exist between core vertices are added to the correction edges list. Core vertices form a super node.

- Neighbors of core vertices are added to the super node if they are connected with edges with at least two halves of the core vertices and if they have at least 1/5 of common neighbors among core vertices with maxVertex.
- Edges between core vertices and added neighbors are removed, and edges that do not exist between core vertices and added neighbors are added to the correction edges list.

In the proposed method, the resulting summary graph contains all remaining edges connecting maximal degree vertices of super nodes.

To optimize the heuristic thresholds for degree-based selection and neighbor similarity, we conducted empirical tuning using the graph datasets described in the next section. The degree-based selection for core vertices is half of the degree of maxVertex. Neighbor vertices with significantly lower degrees than the maxVertex result in many added negative correction edges to the list of correction edges. The degree heuristic threshold is relative, depending on the degree of the maximal degree vertex maxVertex.

The tuning of the similarity threshold of neighbors of core vertices that are assimilated into the super node provided two simple criteria. Neighbors of core vertices must share edges with at least half of the core vertices, and they must maintain a common neighbor threshold of at least 20% with the maxVertex within the core set. Both criteria are relative.

Several papers explicitly note that threshold/parameter spaces are large and that heuristic choices can make results sensitive, which is studied in network thresholding [15].

The proposed ImpZip summarization is a lossless compression method and allows the original data to be perfectly reconstructed.

The method gives summary graphs that preserve important nodes in the graph summary.

The proposed ImpZip summarization method gives summary graphs which are effective for the downstream task of community detection. It is efficient, as many recent community detection methods detect communities by first finding important and influential nodes that become seeds of communities and then growing or refining communities around them using the similarity between nodes. Importance is usually defined via centrality and can be exploited for both accuracy and scalability [16,17].

The ImZip and NI methods explicitly balance model complexity vs. unmodeled edges, so noisy edges are often left in the corrections rather than forcing complex structures, yielding robust coarse summaries [4].

3.2.2. Datasets

Table 1 lists the data sets used in our experiments. These datasets are widely used in the literature of graph summarization [18–20]. The datasets are available from SNAP (http://snap.stanford.edu - *accessed on 2 March 2026*) and LOW (https://law.di.unimi.it/datasets.php -*CNR-2000- accessed on 2 March 2026*).

We remove all edge directions, duplicated edges, and self-loops in the datasets.

3.2.3. Compactness Measure

There are no standardized benchmarking and evaluation practices for graph summarization. Quality of summarization for application-specific tasks such as sampling, aggregation, and visualization can be evaluated using different primary metrics (sampling quality, classification accuracy, visualization clarity, and sparsity) with no cross-task standard [21].

For quantitative comparison, the compactness measure compression ratio is used. Let R be a representation $R = (G_s, C)$ that contains the set of edge corrections C, summary graph G_s with super nodes V_s, and super edges E_s. We evaluate the compactness of R by its relative size to the original edge set E, i.e., $(|E_s| + |C|)/|E|$. $|E_s| + |C|$ is the representation cost. This compactness measure is widely used in the literature of graph

summarization [18–20], except for Slugger. It performs hierarchical graph summarization and uses a different compactness measure. Slugger asks for a representation $RH = (S, P+, P-, H)$ (where S is the set of super nodes, a set H of hierarchy edges between super nodes S, a set P+ of positive edges between super nodes, and a set P− of negative edges between super nodes), and the relative size of the summary graph is $(|P+| + |P-| + |H|)/|E|$ (see [19]).

Table 1. Datasets used in our experiments.

Dataset	N	M	d_avg	Type
Caida	26,475	53,381	4	Internet
Email-Enron	36,692	183,831	10	E-Mail
Brightkite	58,228	214,078	7.4	Geo-Social
Email-Eu-All	265,009	364,481	2.8	E-Mail
Slashdot-0922	82,168	504,230	12.3	Social
DBLP	317,080	1,049,866	6.6	Co-author
Amazon0601	403,394	2,443,408	12.1	Co-purchase
CNR-2000	325,557	2,738,969	16.8	Web
YouTube	1,134,890	2,987,624	5.3	Social
Skitter	1,696,415	11,095,298	13.1	Internet
LiveJournal	3,997,962	34,681,189	17.3	Social

3.2.4. Experimental Results

We compared the ImZip algorithm with Greedy [18], Slugger [19], and LDME [20]. Greedy provides summary compactness on small graphs, while Slugger and LDME are methods that scale to large graphs. The implementation language of all algorithms is Java.

In each experiment, we repeat the method three times and report the average result. Experimental results demonstrate that the proposed NI method achieves compression rates comparable to or slightly lower than traditional Minimum Description Length (MDL) methods across various datasets while maintaining structural integrity.

For eleven test datasets, the ImZip gives lower or equal compactness results than Greedy, LDME, or Slugger (Figure 3). Only for two datasets, the NI method ImZIP gives worse compactness results (*CNR-2000 and Skitter*) than Greedy, LDME, or Slugger. For 4 datasets, it gives better than LDME (*LiveJournal, Amazon 0601, DBLP, and Email-Enron*); for two datasets, it gives better than Slugger (*Caida and Email-Eu-All*), and for two datasets (*Brightkite and YouTube*), it gives a higher compactness measure than LDME and Slugger.

Runtime performances for experimental data are shown in Figures 4 and 5. Figure 5 reports the running time on large graphs. ImZip outperforms LDME by 4.16x on average, and Slugger by 4.24x. On small graphs, ImZip is on average 5.7x faster than Greedy and 2.5x faster than LDME and Slugger (see Figure 4).

The proposed NI method uses local importance that is defined by degree centrality and can be exploited for both accuracy and scalability.

Newer 18 advanced node centrality metrics are analyzed by moving beyond traditional degree centrality [22]. In sparse complex networks of 3 network types, several node centrality metrics run in the order of $O(n\,logn)$. Results show that several centrality measures can cope with large network sizes, which makes them suitable for big network data as well as potential inclusion for composite measures or centrality hybridization [23].

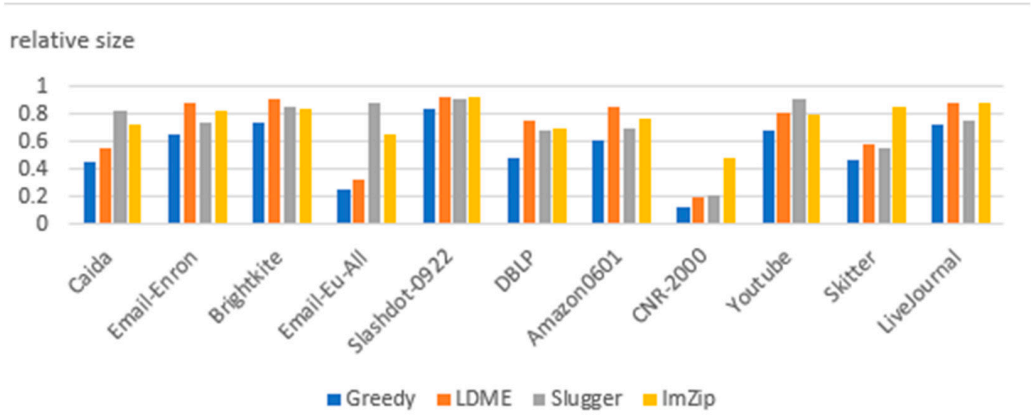


Figure 3. Compactness for 11 datasets of 4 methods: Greedy, LDME, Slugger, and ImZip.

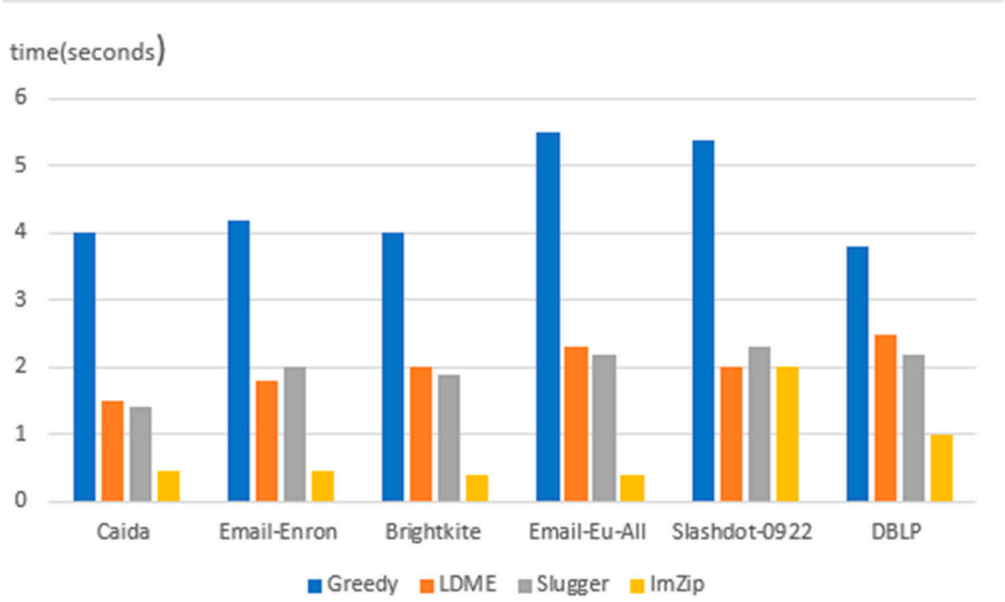


Figure 4. Running time for 6 datasets of 4 methods: Greedy, LDME, Slugger, and ImZip.

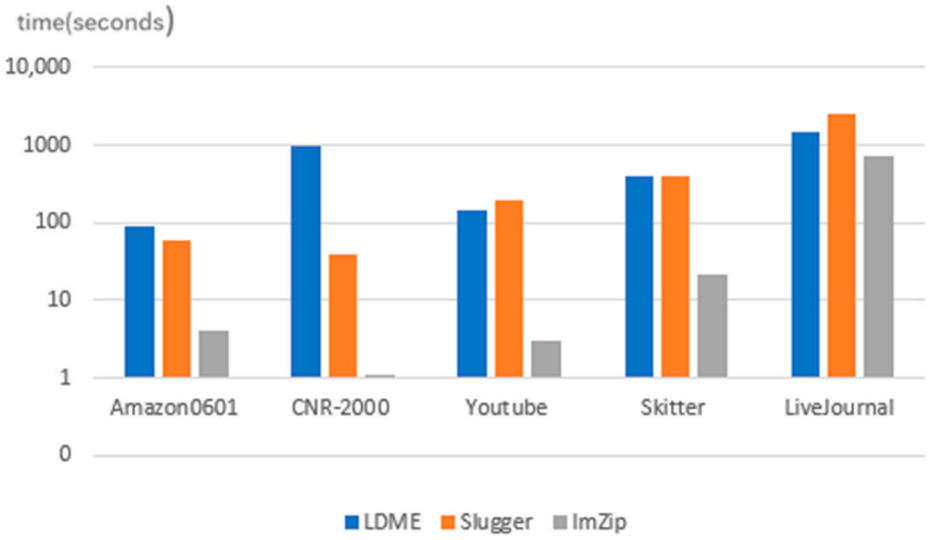


Figure 5. Running time for 5 datasets of 3 methods: LDME and Im, Zip.

4. Graph Neural Networks and Graph Summarization

4.1. GNN Architectures and Graph Summarization

Deep learning (DL) has become a dominant approach in graph summarization because traditional methods (like MDL or basic clustering) become ineffective because of the huge size and high dimensionality of modern network data like huge social networks. DL is now one of the most effective forms of artificial intelligence with its high accuracy. Deep learning techniques capture complex patterns and relationships within the graph. As a deep learning approach, Graph Neural Networks (GNNs) are multi-layer neural networks that learn on graph structures to perform graph-related tasks like summarization, classification, and clustering [24]. GNNs have proved to be a powerful method for graph summarization and as a transformative paradigm in machine learning and artificial intelligence.

Traditional graph summarization approaches are mostly based on conventional machine learning or graph-structured elements, such as degree, adjacency, and eigenvector centrality. Similarity calculations required with traditional approaches are computationally demanding.

Two major advantages of GNNs for graph summarization over traditional methods are the ability to use low-dimensional vectors to represent the features of large graphs and the message-passing mechanism used for communication of information from one node to the next [25]. GNNs are the most successful learning framework for learning the patterns and the subgraphs in large graphs [26].

GNNs can be categorized into more types: Recurrent Graph Neural Networks (RecGNNs), Convolutional Graph Neural Networks (ConvGNNs), Graph Autoencoders (GAEs), and Graph Attention Networks (GATs). Many methods have been proposed for each type of GNN [21].

The notion of GNNs was presented in Gori et al. (2005) [27]. Early studies of GNNs studied RecGNNs.

4.1.1. Recurrent Graph Neural Networks (RecGNNs)

RecGNNs use recurrent neural network architectures to capture temporal dependencies in dynamic graphs [28]. They assume a node in a graph constantly exchanges information/messages with its neighbors and updates node representations iteratively until a stable state is reached. They learn a target node's representation by propagating neighbor information in an iterative manner until a stable fixed point is reached. This is computationally expensive. RecGNNs are effective for tasks like node classification and link prediction.

Node's hidden state is continuously updated by

$$\boldsymbol{h_v^t} = f\Big(\sum_{\boldsymbol{u}\in \boldsymbol{N_v}} \boldsymbol{W}\, \boldsymbol{h_u^{t-1}}\Big) \tag{7}$$

$\boldsymbol{h_v^t}$ is the hidden state of node v at time t, information learned by the RecGNNs about node v at a specific time step in the dynamic graph sequence, $\boldsymbol{N_v}$ denotes the set of neighboring nodes of v, and $\boldsymbol{h_u^{t-1}}$ 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.

$f(\cdot)$ is a simple element-wise activation like ReLU, tanh, or sigmoid.

However, this can be replaced by recurrent update functions, which use gated mechanisms like LSTM (Long Short-Term Memory) [29] or GRU (Gated Recurrent Units) [30] cells.

4.1.2. Convolutional Graph Neural Networks (ConvGNNs)

Convolutional Graph Neural Networks (ConvGNNs) generalize the operation of convolution from grid data to graph data [31]. They generate a node representation by aggregating its own features and neighbors' features. Different from RecGNNs, ConvGNNs stack multiple graph convolutional layers to extract high-level node representations. They apply various weights at each timestep, while RecGNNs apply the same weight matrices in an iterative manner until an equilibrium is reached [32].

ConvGNNs aggregate information of neighbors by a symmetric normalized adjacency matrix A to generate a new node representation. The ConvGNNs model can be described by Equation (8).

$$H = Softmax\Big(\mathrm{A}\ ReLU\ \Big(A\ X\ \boldsymbol{W^{(0)}}\ \Big)\ \boldsymbol{W^{(1)}}\Big) \tag{8}$$

here $\boldsymbol{W^{(0)}}$ and $\boldsymbol{W^{(1)}}$ denote the trainable parameter matrices of the first and second layers of ConvGNNs, respectively. H is the final prediction of the ConvGNNs model. Each ConvGNN layer can aggregate the features of the 1-hop neighborhoods around nodes. However, different weights need to be considered for different nodes in a neighborhood. They aggregate information from a node's neighbors to update its representation. ConvGNNs can be spectral-based [33], using graph signal processing techniques, or spatial-based [34], directly operating on the graph's structure.

4.1.3. Graph Autoencoders (GAEs)

GAEs use an encoder–decoder architecture to learn compact representations of graphs. The encoder transforms the graph into a latent space, and the decoder reconstructs the graph from this representation [35]. GAEs are particularly useful for unsupervised learning tasks like graph clustering.

4.1.4. Graph Attention Networks (GATs)

GATs use attention mechanisms to weigh the importance of different nodes in a neighborhood. They use attention-based neighborhood aggregation, assigning different weights to the nodes in a neighborhood [36]. This allows the model to focus on the most relevant parts of the graph, improving the quality of the summarization. GATs are effective for tasks like node classification and link prediction.

The new representation of the node i is the weighted sum of its neighbors' features.

$$\boldsymbol{h_i^{'}} = \sigma \sum_{j \in \boldsymbol{N_j}} \boldsymbol{a_{ij} W h_j} \tag{9}$$

1. Every node's feature (h_i) is transformed by a shared weight matrix W. This is a linear transformation. h_i and h_j denote the features for the i-th and j-th nodes.
2. Attention coefficients a_{ij} is the weights of the edges between the ith and jth nodes. They are calculated from the following scalar e_{ij} representing the importance of node j to node i: i: $e_{ij} = \ LeakyReLU\left(a^T\ Wh_i\ ||\ Wh_j\right)$ where $||$ denotes concatenation, and a is a weight vector that measures and compares node features.
3. Softmax normalization over weights is performed to make the weights comparable across different neighborhoods. Weights e_{ij} are normalized with the following softmax function:

$$a_{ij} = \frac{\exp\left(\ e_{ij}\right)}{\sum_{k \in N_i} \exp\left(\ e_{ik}\right)} \tag{10}$$

GATs 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, because it reduces storage complexity along with the number of nodes and edges.

4.1.5. Graph Transformers

Unlike GATs, which restrict message passing to immediate neighbors, Graph Transformers propagate representations across all nodes globally, regardless of direct connectivity. This allows them to explicitly model node importance through learned attention mechanisms. Specifically, NIE-GT leverages a Graph-Transformer encoder to capture these long-range dependencies, generating node-importance rankings that outperform classical metrics in aligning with network diffusion and robustness criteria [37].

4.1.6. Robustness to Noisy Graph Data and Scalability of GNNs

In [21], an empirical study and theoretical analysis were conducted on four types of Graph Neural Networks (GNNs), including GCNs and GATs, under three noise conditions: attribute noise, structure noise, and label noise. The results highlight GAT as the most noise-vulnerable architecture and GCL as the most resilient. To address these vulnerabilities, the study introduces a method to improve GAT's noise tolerance.

In GATs, attention can downweight less informative or noisy neighbors, helping focus on relevant parts of the graph, and can capture long-range dependencies and complex structure. However, GATs are often sensitive to sparse, noisy real-world graphs; can overfit on small or noisy datasets; and many GAE/GAT summarization pipelines are weakly regularized, degrading summarization quality under noise or sparsity [38].

Multi-Head Attention GAT improves the expressive power and stabilizes the learning process, making the model more robust to noise.

Scalability is a major challenge for the application of attention-based GNNs in practical scenarios. The most fundamental reason for this phenomenon in attention-based GNNs is the high computational complexity of the attention mechanism. Attention-based GNNs are hard to implement in large graphs, especially for Graph Transformers. The scalability and efficiency of Graph Transformers face significant challenges due to high memory and computational demands, particularly with global attention mechanisms for large-scale graphs. To mitigate these issues, several strategies are proposed, such as applying graph sparsification, using graph partitioning, or using sampling methods [39].

Up to now, although some approaches have attempted to improve model efficiency through sampling and subgraphs, they still cannot efficiently process large graphs. Many scalable methods trade some global expressivity (e.g., via sampling or anchors), while those keeping global attention must rely on aggressive simplification or hardware support [40].

4.2. Node Importance Summarization Using GATs

If efficiency is primary and summaries are mostly structural, ConvGNNs can be used, while for fine-grained node-importance-driven summaries, GATs are the most efficient for learning importance weights explicitly. GATs perform graph summarization using node importance, which involves transforming complex graphs into condensed, informative versions. In contrast to traditional summarization, GATs use attention to ensure that the summary graph retains the most structurally and semantically significant components. The core goal of GATs is to assign different importance (weights) to different nodes in a neighborhood, allowing the model to focus on the most relevant features for a given task.

5. GAT Models

GATs are Graph Neural Networks that utilize an attention mechanism to prioritize specific neighbors when aggregating information from a network. A comprehensive survey of methods, applications, advantages, and limitations of the GAT model is in [41,42].

GATs have evolved into several specialized categories, each tailored to address distinct challenges in graph-structured data processing.

The GATv2 model computes dynamic attention and can represent much more complex relationships. It can learn a different ranking of neighbors by node importance, depending on which node is looking at them.

Multi-view GATs allow a model to weigh multiple distinct types of relationships simultaneously, each in its own view. It processes these different views to create a more complete understanding of a node and its importance.

Multi-relational GATs do not treat all edges as identical, as standard GATs do when a graph contains multiple types of relationships (e.g., in a Knowledge Graph).

Sparse GATs are tailored to solve the density problems of GATs that calculate attention for every edge in the graph, which is computationally expensive in large-scale networks. They make the attention mechanism more selective and efficient.

Deeper/multi-layer GATs are a new version of GATs, which can effectively alleviate the oversmoothing issue of GATs. As layer depth increases in GATs, they often fail to regulate excessive propagation from dominant nodes, leading to oversmoothing.

Graph Topology Attention Networks extracts topology features from the graph's structure and integrates the node and topology representations through cross-attention GNN layers. They can dynamically balance node features with structural data and can enhance graph summarization.

This session describes the advantages and disadvantages of GATs and some methods of improved GAT architecture.

5.1. Advantages and Disadvantages of GATs

GATs have many advantages: selective weighting, computational efficiency, and interpretability.

GATs provide selective weighting and learn data-dependent weights, unlike older network models that treat every neighbor equally or use fixed weights based on the graph structure. It can ignore noisy neighbors and focus on those that are most relevant to the task.

GATs are computationally efficient. The attention mechanism can be calculated in parallel across all edges in the graph. It does not require expensive matrix inversions or global knowledge of graph structure. The time complexity of a single GAT attention head computing F features may be expressed as $O(n\ F_i\ F + m\ F)$, where F_i is the number of input features, and n and m are the numbers of nodes and edges in the graph, respectively. This complexity is on par with ConvGNNs [31].

Applying multi-head attention multiplies the storage and parameter requirements by a factor of K, while the individual heads' computations are fully independent and can be parallelized. Interpretability is provided with attention coefficients, which act as an indicator of importance. You can visualize these weights to see which specific nodes care about when deciding. It naturally handles nodes with different degree distributions (number of neighbors) because the attention scores are normalized locally. But GATs are memory-intensive, especially with multi-head attention, while they are computationally fast. Storing attention scores for every edge in a very large graph requires significant GPU memory compared to simpler models.

The GAT model uses a static form of attention. This means the ranking of neighbors is the same for every query node. It lacks the query-dependent ranking found in models like Transformers. One disadvantage is oversmoothing in deep networks. Like most GNNs, if you stack too many GAT layers, the node representations tend to become identical, converging to a mean value, which diminishes their accuracy. GATs are sensitive to noise.

Despite its robustness against noisy neighbors, the model is highly dependent on the precision of its starting node features. If the features are poor, the attention mechanism may fail to learn meaningful weights. GATs are very complex for small graphs. For very simple or small datasets, the added complexity of the attention mechanism often does not provide enough benefit over standard GNN to justify the extra parameters.

5.2. GATv2

Brody et al. (2021) proved that the original GAT model calculates static attention and introduced an improved version of GAT called GATv2 [43]. In the GAT model, the ranking of importance among neighbors is the same for every query node. If one neighbor node v is more important than the other neighbor node u, it will be more important for all nodes in the graph. Static attention is when the attention to the key nodes has the same rank (order) for any query node. GAT computes attention using a scalar e_{ij} representing the importance of node j to node i:

$$e_{ij} = a^T LeakyReLU\left(Wh_i||Wh_j\right) = LeakyReLU\left(a_1^T\ Wh_i||a_2^T\ Wh_j\right) \tag{11}$$

where $||$ denotes concatenation, and a is a weight vector that measures and compares node features. Note that for any query node, the attention rank of keys depends only on $a_2^T\ Wh_j$. Therefore, the attention range of keys is static; it remains the same for all queries. GATv2 allows dynamic attention by changing the attention mechanism so that it uses a scalar e_{ij} (Equation (12)) for representing the importance of node j to node i to

$$e_{ij} = a^T\ LeakyReLU\left(W_1\ h_i||W_2h_j\right) \tag{12}$$

This allows the model to learn a different ranking of neighbors depending on which node is looking at them. GATv2 can represent much more complex relationships. GAT can be replaced with GATv2 in any architecture and usually yields better results without adding complexity. For very simple graphs where a static ranking is good enough, GATv2 might not show a significant improvement over the original GAT. GATv2, like the original GAT, suffers from oversmoothing when you add too many layers. GATv2 has high memory consumption due to storing attention weights for every edge.

5.3. Multi-View Graph Attention Networks

Real-world networks often feature nodes with diverse connection types, representing different layers of interaction within the same system. For example, in a social network, two people might be friends or colleagues at work. Each of these is a different view of the same graph. Multi-view GAT (MGAT) has been proposed, which has multiple graphs (views) as input, while GAT has a single graph as input [44]. MGAT splits a heterogeneous graph with several relationships into single-view graphs and then integrates information from multiple views into the embedding of nodes. MGAT uses node-to-node attention, also known as view-to-view attention. It captures the complexity of heterogeneous data.

The MGAT architecture works in two stages. First, for each individual view-graph layer, the model applies a standard GAT mechanism. This allows the model to learn the importance of neighbors within that specific context. Second, after processing each view, the model uses another attention mechanism to weigh the views themselves. If the friendship view is more relevant for predicting a user's social life than the professional view, the model assigns it a higher weight. The model automatically identifies which view is most informative for a specific task. This acts as an automated feature selection process. MGATs are more robust to noise. If one view is noisy or has missing edges, the model can rely on other views to compensate, leading to more stable predictions.

MGAT runs multiple GAT layers in parallel, each layer for each view, which increases the number of parameters and the time required for training. The memory requirements grow linearly with the increase in views. Therefore, it is difficult to apply to massive datasets with many different relationship types. In MGAT, nodes are the same across all views. If the views have different sets of nodes, the alignment of these views becomes difficult.

An example of an application is a proposed multi-view graph attention network for travel recommendations [45]. The model takes several different types of user behaviors into account, such as making hotel reservations, booking flights, and learning an attention mechanism to weigh the importance of different views for a recommendation.

5.4. Multi-Relational Graph Attention Networks

GNNs are popular models that can exploit and use the structural information of neighbors in knowledge graphs (KGs). Traditional GATs often have problems with KGs, because they are defined by the types of relations between those nodes. It uses an attention mechanism that specifically processes the type of relation connecting two entities. Multi-Relational GAT (MRGAT) can adapt to different cases of heterogeneous multi-relational connections and then calculate the importance of different neighboring nodes through a self-attention layer [46]. The use of a self-attention mechanism with different node weights optimizes the network structure and increases performance.

5.5. Sparse GATs

Ye et al. proposed a framework that learns sparse attention coefficients [47]. It simplifies GAT by assigning a single attention coefficient per edge across layers, effectively pruning 50–80% of edges in large graphs without losing accuracy.

5.6. Deeper/Multi-Layer GATs

GATs and ConvGNNs are two state-of-the-art GNNs. Both architectures suffer from performance degradation when more GNN layers are stacked. With the increase in layer depth in GATs, they often fail to regulate excessive propagation from dominant nodes, leading to oversmoothing. A new version of GAT named Layer-wise Self-adaptive GAT (LSGAT), which can effectively alleviate the oversmoothing issue in deep GAT, has been proposed [48]. By redesigning the attention mechanism to adaptively adjust based on layer depth, LSGAT considers both immediate neighboring and non-adjacent nodes from a global view. This approach effectively mitigates oversmoothing and is more expressive than the original GAT architecture. It is more effective with better results on node classification tasks.

5.7. Graph Topology Attention Networks (GTAT)

GNNs perform representation learning on graph-structured data and do not integrate topological information into graph representation learning. They only capture the information of nodes by recursively aggregating the neighboring nodes' representations. To better integrate topological information, Graph Topology Attention Networks (GTATs) extract topology features from the graph's structure and encode them into topology representations [49]. The node and topology representations are integrated through cross-attention GNN layers. The GTAT model can dynamically balance node features with structural data and can enhance graph summarization.

6. Extensions of the GAT Notion of Importance

In GATs, each node learns attention weights over its neighbors. Attention coefficients quantify how important each neighbor is for updating the node's representation. The weights depend on neighbor features and shared parameters and are normalized with the

softmax function. Attention weights are learned, a feature-aware metric for local node importance, enriching degree-based normalization typically found in standard ConvGNNs.

GATs define importance via learned, local attention weights over neighbors. Many variants extend this notion along different axes: locality vs. globality, dynamics, relation types, and sparsity. Extensions of the GAT notion of importance are also performed by the methods using higher-order neighborhoods to capture multi-hop influence, methods combining node content and structure to get more faithful importance scores, and GAT models making attention dynamic (e.g., GATv2, LSGAT), so rankings depend on the query node.

6.1. Methods Using Higher-Order Neighborhoods to Capture Multi-Hop Influence

Many extensions of the GAT architecture explicitly incorporate higher-order neighborhoods to capture multi-hop influence while mitigating oversmoothing and over-squashing.

GATs only focus on low-order content, while high-order content is completely ignored. High-Order GAT (HGRN) aggregates features from neighbors at multiple hop distances with attention and iteratively updates graph topology to better reflect high-order relations to mitigate oversmoothing and to improve node classification [50].

Multi-hop Attention Graph Neural Network (MAGNA) incorporates multi-hop context information into every layer of attention computation [51]. It introduces multi-hop attention diffusion: attention scores are diffused over the adjacency matrix, so a single layer can attend to multi-hop neighbors, efficiently including all paths and improving long-range structural encoding. By using a diffusion prior on attention values, MAGNA captures the influence of all underlying paths between disconnected nodes without affecting efficiency.

A Multi-Order-Content-Based Adaptive Graph Attention Network for graph node classification has been proposed [52]. It constructed a high-order content attention mechanism that could focus on high-order content to evaluate attention weights. The framework also integrates multi-order attention with a hybrid approach that captures the interplay between high- and low-order content.

Multi-Head Multi-Order Graph Attention Networks (MHMOGAT) is an enhanced GAT layer [53]. MHMOGAT leverages multi-head attention alongside multi-order adjacency matrices to extend the receptive field of traditional GATs. This architecture enables the model to capture long-range dependencies across the graph more effectively. To ensure peak performance across diverse datasets, Bayesian optimization is used to identify the ideal hyperparameter configurations.

Layer-wise Self-adaptive GAT (LSGAT) effectively mitigates the oversmoothing problem common in deep GATs and improves node classification [48]. By adaptively adjusting the attention coefficient mechanism based on layer depth, LSGAT integrates both immediate neighbors and non-adjacent nodes through a global perspective.

6.2. GATs Combining Node Content and Structure

Methods combining node content and structure to get more faithful importance scores have been proposed.

Content- and Structure-based Graph Attention Network (CSGAT) has been proposed, which uses a multi-head attention mechanism to define attention weights by integrating node content with topological data [54]. CSGAs incorporate structural interactions into the standard GAT framework. It captures higher-order structural information without requiring additional training, and its results are interpretable via heatmap visualization.

GATs only use the topology of edges to extract network information, while the associations between node features are not used. The proposed Semi-Supervised Multi-Channel Attention Network (MC-GAT) simultaneously extracts node features and topological struc-

tures [55] using the attention mechanism that can automatically assign different weights to different pieces of information representing different aspects. The attention mechanism is utilized to assign specific weights to each informational channel, ensuring that various aspects of the data are integrated according to their importance.

7. Using GNNs for Node Importance Estimation

Several works explicitly extract node importance or influence from attention or related GNN modules.

7.1. General Node Centrality

Graph Attention-based Network Representation (GANR) is a framework that leverages graph attention architectures by using structural topology as a supervised learning input [56]. GANR learns high-quality node representations for efficient link prediction, network visualization, and node classification, and extracts meaningful attention weights that can be used for node centrality measures.

7.2. Knowledge Graph Node Importance

GENI is a GNN-based method designed to deal with predicting node importance in knowledge graphs [57]. It performs an aggregation of importance scores instead of aggregating node embeddings via a predicate-aware attention mechanism and flexible centrality adjustment.

7.3. Dynamic-Network Node Ranking

In [58], a new node importance prediction method is proposed that uses community detection, joint embeddings, multi-layer GAT, and then applies contrastive learning to emphasize multi-scale structural differences in dynamic networks. It produces node ranking for dynamic networks.

Community detection is employed to extract node-level community features. The joint embedding module is used for incorporating community information into node representations. A multi-layer GAT adaptively learns node and neighborhood features.

8. Application-Specific GAT Research

Attention weights or attention-based pooling layers naturally produce node rankings, allowing the selection of top-k important nodes as a graph summary, building coarser graphs from high-attention nodes and edges, and task-specific summaries such as text or code summarization, where GAT focuses on key nodes and yields better summaries than non-attentional methods.

Coarser graphs from high-attention nodes and edges can be built with Graph Attention-based Network Representation (GANR) [56]. It utilizes a graph attention architecture. GANR learns high-quality node representations and performs efficient link prediction, network visualization, and node classification.

GATs focus on key nodes and produce better task-specific summarization, such as text or code summarization, than non-attentional methods. GATs are widely used to compress graphs or graph-structured content (text, code, documents, videos) into compact representations while preserving important structure and semantics.

For text summarization, the Gated Attention Graph Neural Network (GA-GNN) based on gated attention has been proposed [59]. It improves the accuracy and readability of text summarization. A parallel mechanism is used to encode sentences into a deeper vector containing local and global semantic information. After that, sentence vectors are converted into graph structure information for feature extraction. Gated attention units are

introduced into the model, which are used in combination with Graph Neural Networks to eliminate redundant information and improve model performance. The loss function is optimized from contrast learning, important sentence confidence calculation, and Graph Neural Networks.

For source code summarization that generates semantic and structural information descriptions for Java methods in a natural language, a novel architecture, GSCS, has been proposed [60]. GATs are used to process the tokenized abstract syntax tree of the program. Using multi-head attention, the model identifies diverse relational patterns in various subspaces. The model then integrates these features through a weighted aggregation across the node's local neighborhood.

In [61], a framework named GTASum is proposed. It uses GAT to capture relationships between sentences through the document representation of graph structure and simultaneously updates the local and global information.

Some research focuses on structural summarization and how to use attention to sample and condense massive graphs for faster retrieval without losing connectivity information. A framework has been proposed that merges attention-based graph summarization with state-of-the-art graph sampling methods tailored explicitly for large-scale graph processing and information retrieval applications [62].

9. Conclusions

This study investigates a node importance (NI) approach for graph summarization, shifting the focus from mere size reduction to the preservation of structurally significant entities.

We proposed a simple NI method and compared the compactness ratio of graph summarization with popular MDL methods. The results show that the NI approach yields significant compression ratios for most datasets in our experiment.

Because traditional methods like MDL or basic clustering often struggle with the scale and high dimensionality of modern network data, deep learning with Graph Neural Networks (GNNs) has been investigated for graph summarization. This study investigates the structure of key GNN architectures, focusing on key architectures such as RecGNN, ConvGNN, GAE, and GAT, and their suitability in the context of graph summarization using the NI approach. GATs and their variants are a promising architecture that provides a flexible, learned notion of node importance via attention.

GATs have evolved into several specialized architectures, each tailored to address distinct challenges in graph-structured data processing, such as multi-view GATs, multi-relational GATs, sparse GATs, deeper/multi-layer GATs, and Graph Topology Attention Networks.

We discuss extensions of the GAT notion of importance performed by the methods using higher-order neighborhoods to capture multi-hop influence, methods combining node content and structure to get more faithful importance scores, and GAT models making attention dynamic and making rankings depend on the query node.

Several research works explicitly extract node importance or influence from attention or related GNN modules. We discuss general node centrality, knowledge graph node imp3ortance, and dynamic-network node ranking.

Attention weights or attention-based pooling layers naturally induce node rankings, allowing for selecting the top k important nodes as a graph summary or building coarser graphs from high-attention nodes and edges. We also discuss some methods for task-specific summaries for text or code summarization, where GAT focuses on key nodes and yields better summaries than non-attentional methods. While the original GAT introduced the concept of learned importance for neighbor nodes, the next generation of GAT research

has to solve the limitations of the original GAT architecture: scalability, dynamic data, and explainability.

Author Contributions: Conceptualization, K.R.Ž. and D.M.; methodology, K.R.Ž. and M.Ž., software, K.R.Ž.; investigation, K.R.Ž. and M.Ž.; writing—original draft preparation, K.R.Ž. and M.Ž., writing—review and editing, K.R.Ž., M.Ž. and D.M.; visualization, K.R.Ž.; funding acquisition, D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Slovene Research Agency under Research Program P2-0041.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GNN	Graph Neural Network
RecGNN	Graph recurrent neural network
ConvGNNs	Convolution Graph Neural Networks
GAT	Graph attention network
GAEs	Graph autoencoders

References

1. Meta—Meta Reports Fourth Quarter and Full Year 2024 Results. Available online: https://investor.atmeta.com/investor-news/press-release-details/2025/Meta-Reports-Fourth-Quarter-and-Full-Year-2024-Results/ (accessed on 2 March 2026).
2. Aggarwal, C.C.; Wang, H. *Managing and Mining Graph Data*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 40.
3. Bonifati, A.; Dumbrava, S.; Kondylakis, H. Graph Summarization. *arXiv* **2020**, arXiv:2004.14794. [CrossRef]
4. Liu, Y.; Safavi, T.; Dighe, A.; Koutra, D. Graph summarization methods and applications: A survey. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–34. [CrossRef]
5. Song, C.; Havlin, S.; Makse, H. Self-similarity of complex networks. *Nature* **2005**, *433*, 392–395. [CrossRef]
6. Albert RJeong, H.; Barabasi, A.-L. Diameter of the World Wide Web. *Nature* **1999**, *401*, 130–131. [CrossRef]
7. Wang, X.; Chen, G. Complex networks: Small-world, scale-free and beyond. *IEEE Circuits Syst. Mag.* **2003**, *3*, 6–20. [CrossRef]
8. Kumar, K.A.; Efstathopoulos, P. Utility-driven graph summarization. *Proc. VLDB Endow.* **2018**, *12*, 335–347. [CrossRef]
9. Hashemi, M.; Gong, S.; Ni, J.; Fan, W.; Prakash, B.A.; Jin, W. A comprehensive survey on graph reduction: Sparsification, coarsening, and condensation. *arXiv* **2024**, arXiv:2402.03358. [CrossRef]
10. Lakshmanan, L.V.; Ng, R.T.; Wang, C.X.; Zhou, X.; Johnson, T.J. The generalized MDL approach for summarization. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*; Morgan Kaufmann: Burlington, MA, USA, 2002; pp. 766–777.
11. Jin, W.; Zhao, L.; Zhang, S.; Liu, Y.; Tang, J.; Shah, N. Graph condensation for graph neural networks. *arXiv* **2021**, arXiv:2110.07580.
12. Boldi, P.; Vigna, S. The webgraph framework I: Compression techniques. In *Proceedings of the 13th International Conference on World Wide Web*; ACM Press: New York, NY, USA, 2004; pp. 595–602.
13. Chiranjeevi, M.; Dhuli, S.; Enduri, M.; Hajarathaiah, K.; Cenkeramaddi, L. Quantifying Node Influence in Networks: Isolating-Betweenness Centrality for Improved Ranking. *IEEE Access* **2024**, *12*, 93711–93722. [CrossRef]
14. Ullah, A.; Wang, B.; Sheng, J.; Long, J.; Khan, N.; Sun, Z. Identification of nodes influence based on global structure model in complex networks. *Sci. Rep.* **2021**, *11*, 6173. [CrossRef] [PubMed]
15. Schroeder, A.; Funk, R.; Guan, J.; Okonek, T.; Ziegelmeier, L. Higher-Order Network Structure Inference: A Topological Approach to Network Selection. *arXiv* **2025**, arXiv:2510.04884. [CrossRef]
16. Aghaalizadeh, S.; Afshord, S.; Bouyer, A.; Anari, B. A three-stage algorithm for local community detection based on the high node importance ranking in social networks. *Phys. A-Stat. Mech. Its Appl.* **2021**, *563*, 125420. [CrossRef]
17. Sheykhzadeh, J.; Zarei, B.; Gharehchopogh, F. Community Detection in Social Networks Using a Local Approach Based on Node Ranking. *IEEE Access* **2024**, *12*, 92892–92905. [CrossRef]
18. Navlakha, S.; Rastogi, R.; Shrivastava, N. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*; ACM Press: New York, NY, USA, 2008; pp. 419–432.
19. Lee, K.; Ko, J.; Shin, K. Slugger: Lossless hierarchical summarization of massive graphs. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*; IEEE: New York, NY, USA, 2022; pp. 472–484.

20. Yong, Q.; Hajiabadi, M.; Srinivasan, V.; Thomo, A. Efficient graph summarization using weighted lsh at billion-scale. In *Proceedings of the 2021 International Conference on Management of Data*; ACM Press: New York, NY, USA, 2021; pp. 2357–2365.
21. Shabani, N.; Wu, J.; Beheshti, A.; Sheng, Q.; Foo, J.; Haghighi, V.; Hanif, A.; Shahabikargar, M. A Comprehensive Survey on Graph Summarization With Graph Neural Networks. *IEEE Trans. Artif. Intell.* **2023**, *5*, 3780–3800. [CrossRef]
22. Freund, A.J.; Giabbanelli, P.J. An experimental study on the scalability of recent node centrality metrics in sparse complex networks. *Front. Big Data* **2022**, *5*, 797584. [CrossRef]
23. Singh, A.; Singh, R.R.; Iyengar, S.R.S. Node-weighted centrality: A new way of centrality hybridization: A. Singh et al. *Comput. Soc. Netw.* **2020**, *7*, 6. [CrossRef]
24. Khoshraftar, S.; An, A. A survey on graph representation learning methods. *ACM Trans. Intell. Syst. Technol.* **2024**, *15*, 1–55. [CrossRef]
25. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
26. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
27. Gori, M.; Monfardini, G.; Scarselli, F. A new model for learning in graph domains. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*; IEEE: New York, NY, USA, 2005; Volume 2, pp. 729–734.
28. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [CrossRef] [PubMed]
29. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
30. Cho, K.; Van Merriënboer, B.; Gulçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Proceedings of the 2014 Conference On Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
31. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2017**, arXiv:1609.02907. [CrossRef]
32. Zhang, S.; Tong, H.; Xu, J.; Maciejewski, R. Graph convolutional networks: A comprehensive review. *Comput. Soc. Netw.* **2019**, *6*, 11. [CrossRef] [PubMed]
33. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv* **2013**, arXiv:1312.6203.
34. Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Trans. Neural Netw.* **2009**, *20*, 498–511. [CrossRef]
35. Pinaya, W.H.L.; Vieira, S.; Garcia-Dias, R.; Mechelli, A. *Autoencoders, in Machine Learning*; Academic Press: San Diego, CA, USA, 2020; pp. 193–208.
36. Shehzad, A.; Xia, F.; Abid, S.; Peng, C.; Yu, S.; Zhang, D.; Verspoor, K. Graph Transformers: A Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2025**, *early access*. [CrossRef]
37. Jin, Y.; Zhu, X. A Systematic Study and Analysis of Graph Neural Networks under Noise. *ACM Trans. Knowl. Discov. Data* **2025**, *19*, 1–20. [CrossRef]
38. Sun, C.; Li, C.; Lin, X.; Zheng, T.; Meng, F.; Rui, X.; Wang, Z. Attention-based graph neural networks: A survey. *Artif. Intell. Rev.* **2023**, *56*, 2263–2310. [CrossRef]
39. Zhu, W.; Song, G.; Wang, L.; Liu, S. Anchorgt: Efficient and flexible attention architecture for scalable graph transformers. *arXiv* **2024**, arXiv:2405.03481. [CrossRef]
40. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
41. Chaudhari, S.; Mithal, V.; Polatkan, G.; Ramanath, R. An attentive survey of attention models. *ACM Trans. Intell. Syst. Technol. (TIST)* **2021**, *12*, 1–32. [CrossRef]
42. Vrahatis, A.G.; Lazaros, K.; Kotsiantis, S. Graph Attention Networks: A Comprehensive Review of Methods and Applications. *Future Internet* **2024**, *16*, 318. [CrossRef]
43. Brody, S.; Alon, U.; Yahav, E. How attentive are graph attention networks? *arXiv* **2021**, arXiv:2105.14491.
44. Xie, Y.; Zhang, Y.; Gong, M.; Tang, Z.; Han, C. Mgat: Multi-view graph attention networks. *Neural Netw.* **2020**, *132*, 180–189. [CrossRef]
45. Chen, L.; Cao, J.; Wang, Y.; Liang, W.; Zhu, G. Multi-view graph attention network for travel recommendation. *Expert Syst. Appl.* **2022**, *191*, 116234. [CrossRef]
46. Dai, G.; Wang, X.; Zou, X.; Liu, C.; Cen, S. MRGAT: Multi-relational graph attention network for knowledge graph completion. *Neural Netw.* **2022**, *154*, 234–245. [CrossRef] [PubMed]
47. Ye, Y.; Ji, S. Sparse graph attention networks. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 905–916. [CrossRef]
48. Su, G.; Wang, H.; Zhang, Y.; Zhang, W.; Lin, X. Simple and deep graph attention networks. *Knowl.-Based Syst.* **2024**, *293*, 111649. [CrossRef]
49. Shen, J.; Ain, Q.T.; Liu, Y.; Liang, B.; Qiang, X.; Kou, Z. GTAT: Empowering graph neural networks with cross attention. *Sci. Rep.* **2025**, *15*, 4760. [CrossRef]

50. He, L.; Bai, L.; Yang, X.; Du, H.; Liang, J. High-order graph attention network. *Inf. Sci.* **2023**, *630*, 222–234. [CrossRef]
51. Wang, G.; Ying, R.; Huang, J.; Leskovec, J. Multi-hop attention graph neural network. *arXiv* **2020**, arXiv:2009.14332.
52. Chen, Y.; Xie, X.-Z.; Weng, W.; He, Y.-F. Multi-Order-Content-Based Adaptive Graph Attention Network for Graph Node Classification. *Symmetry* **2023**, *15*, 1036. [CrossRef]
53. Ben, J.; Sun, Q.; Liu, K.; Yang, X.; Zhang, F. Multi-head multi-order graph attention networks. *Appl. Intell.* **2024**, *54*, 8092–8107. [CrossRef]
54. Chen, Y.; Xie, X.; Weng, W. Content and structure-based attention for graph node classification. *J. Intell. Fuzzy Syst.* **2024**, *46*, 8329–8343. [CrossRef]
55. La, Z.; Qian, Y.; Leng, H.; Gu, T.; Gong, W.; Chen, J. MC-GAT: Multi-Channel Graph Attention Networks for Capturing Diverse Information in Complex Graphs. *Cogn. Comput.* **2023**, *16*, 595–607. [CrossRef]
56. Gu, W.; Gao, F.; Lou, X.; Zhang, J. Discovering latent node Information by graph attention network. *Sci. Rep.* **2021**, *11*, 6967. [CrossRef]
57. Park, N.; Kan, A.; Dong, X.L.; Zhao, T.; Faloutsos, C. Estimating node importance in knowledge graphs using graph neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*; ACM Press: New York, NY, USA, 2019; pp. 596–606.
58. Ai, J.; Zhang, Y.; Su, Z.; Guo, C.; Li, M. A Node Importance Prediction Algorithm Based on Graph Attention and Contrastive Learning. *Chin. Phys. B* **2025**. [CrossRef]
59. Huang, J.; Wu, W.; Li, J.; Wang, S. Text summarization method based on gated attention graph neural network. *Sensors* **2023**, *23*, 1654. [CrossRef]
60. Zhou, Y.; Shen, J.; Zhang, X.; Yang, W.; Han, T.; Chen, T. Automatic source code summarization with graph attention networks. *J. Syst. Softw.* **2022**, *188*, 111257. [CrossRef]
61. Jiang, M.; Zou, Y.; Xu, J.; Zhang, M. GATSum: Graph-based topic-aware abstract text summarization. *Inf. Technol. Control.* **2022**, *51*, 345–355. [CrossRef]
62. Shabani, N.; Beheshti, A.; Jolfaei, A.; Wu, J.; Haghighi, V.; Najafabadi, M.K.; Foo, J. Attention-based graph summarization for large-scale information retrieval. *IEEE Trans. Consum. Electron.* **2024**, *70*, 6224–6235. [CrossRef]