Cluster-Aware Graph Summaries through Graph Matching Neural Networks

Achilleas Tsimichodimos*, Angeliki Dimitriou*, Nikolaos Chaidos*, Giorgos Stamou*
*School of Electrical and Computer Engineering
National Technical University of Athens
Athens, Greece

***Abstract*—As the size and complexity of graph-structured data increase, graph clustering has become an essential task in analyzing complex networks across various domains. This work explores the application of Graph Neural Networks in graph clustering, comparing them with traditional methods like Spectral Clustering to understand their effectiveness in creating meaningful graph summaries. Additionally, we investigate the potential of Graph Matching Networks to improve these GNN-based methods. We utilize GMNs both as an evaluation metric and as part of the loss function during training, aiming to enhance clustering results by considering the overall structural integrity of the graphs.**

***Index Terms*—Graph Neural Networks, Graph Matching Networks, Graph Clustering Graph Summarization**

I. Introduction

The analysis of graph-structured data has become increasingly important across various domains, including social networks [15], molecular biology [3], and recommender systems [18]. With the escalating size and complexity of these graphs, graph summarization [11] is gaining prominence, as it offers several advantages, including accelerating graph-related tasks, enhancing accuracy [4], and improving explainability, by offering a promising avenue for identifying meaningful patterns and subgraphs that can be leveraged for insightful explanations. However, the high-dimensional nature and complexity of these graphs pose significant challenges. Traditional methods for graph clustering and summarization often struggle to capture intricate structures, particularly when dealing with complex, non-linear relationships within the data.

In recent years, Graph Neural Networks (GNNs) have achieved state-of-the-art performance in various graph analysis tasks, such as node classification and link prediction, and have surged as a potent solution, offering advanced techniques to address these challenges by leveraging both node features and graph topology.

In this study, we examine a spectrum of graph clustering methods, focusing on techniques that utilize GNNs and are based on graph pooling (MinCutPooling, DMoN, JustBalance Pooling) or embedding clustering (Variational Graph Auto-Encoders). We also explore Node2Vec, another embedding-based approach. Our investigation includes a comparative analysis with traditional techniques such as Spectral Clustering.

Furthermore, this study introduces the integration of Graph Matching Networks (GMNs) with GNN-based clustering methods. GMNs, known for their capability to assess graph similarity, are used in two ways: as an auxiliary metric to evaluate the clustering performance and as part of the loss function during the training process. This novel approach aims to enhance the quality of graph clustering by considering not only the node and graph features but also the structural integrity of the graphs.

Through this work, we seek to provide insights into the efficacy of different GNN-based clustering techniques and the potential benefits of incorporating GMNs in the training and evaluation process.

II. Background

Let a graph be represented by a tuple $G = \{\mathcal{V}, \mathcal{E}\}$, with node set $\mathcal{V}$ and edge set $\mathcal{E}$. Let $|\mathcal{V}| = N$ and $|\mathcal{E}| = E$ be the number of nodes and edges, respectively. Each node i is associated with a feature vector $\boldsymbol{x}_i \in \mathbb{R}^{\mathbb{F}}$. A graph is described by its adjacency matrix $A \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ and the node features matrix $X \in \mathbb{R}^{\mathbb{N}\times\mathbb{F}}$.

A. Graph Clustering

Graph clustering is the process of grouping nodes in a graph based on their interconnections and similarities. This method is employed to identify underlying structures within graphs, such as communities or closely linked groups. A variety of methods are used for graph clustering, including spectral clustering, which uses the eigenvalues of the graph Laplacian, and approaches that employ Graph Neural Networks (GNNs) for node embedding. The selection of a clustering technique depends on the graph's nature and the specific goals of the analysis. The key challenge in graph clustering is to effectively capture the complexities of node relationships and graph topology, balancing accuracy with computational demands.

B. Graph Neural Networks

Graph Neural Networks are a flexible class of models that perform nonlinear feature aggregation with respect to graph structure.

The core operation in a GNN can be mathematically represented as follows:

$$h_v^{(l+1)} = \text{UPDATE}^{(l)}\Big(h_v^{(l)}, \text{AGGREGATE}^{(l)}\left(\{A_{vu} \cdot h_u^{(l)} : u \in \mathcal{N}(v)\}\right)\Big), \tag{1}$$

where $h_v^{(l)}$ represents the feature vector of node v at layer l, extracted from the feature matrix X. The adjacency matrix A

is used in the AGGREGATE function to weigh the influence of neighboring nodes' features, denoted by $h_u^{(l)}$, based on their connectivity. The UPDATE function then combines these aggregated features to update the node's representation.

GNNs are applied in tasks such as node classification, link prediction, and graph-based clustering, making use of the enriched node representations that account for both individual node attributes and the broader graph structure.

C. Embedding-based Approaches for Graph Clustering

1) Spectral Clustering: Spectral Clustering [14], [17] is one of the traditional graph clustering approaches, which uses the eigenvalues of the Laplacian matrix for dimensionality reduction prior to clustering. Depending on the specific application, various forms of the Laplacian are employed, such as the unnormalized Laplacian $L = D - A$, the normalized Laplacian $L_{\text{norm}} = D^{-1/2}LD^{-1/2}$, and the random walk normalized Laplacian $L_{\text{rw}} = D^{-1}L$. In these expressions, A denotes the adjacency matrix, and D represents the diagonal degree matrix.

2) Node2Vec: Node2Vec [6] is an algorithm for learning feature representations of nodes in a graph. It extends the Skip-Gram model [12] to graph data by performing biased random walks, effectively capturing the diversity of connectivity patterns in a graph. The method balances the exploration of local neighborhoods (like Breadth-First Search) and distant parts of the graph (similar to Depth-First Search) through its random walks.

The formal representation of Node2Vec's random walk process can be described as follows: Given a source node u, Node2Vec samples a fixed-length random walk path with a transition probability between nodes u and v based on their structural proximity. The transition probability is influenced by parameters p and q, which dictate the likelihood of immediately revisiting a node and exploring nodes further away, respectively.

3) VGAE: Variational Graph Auto-Encoders (VGAE) [8] are a framework for unsupervised learning on graph-structured data, utilizing the principles of variational autoencoders [7]. VGAE employs a graph convolutional network (GCN) [9] as an encoder to map nodes to a latent space, representing each node with a low-dimensional vector. The VGAE decoder reconstructs the graph's adjacency matrix from these latent representations, allowing for the generation of new graph structures or the prediction of missing links. The VGAE's objective function comprises two primary components: the reconstruction loss and the regularization term. The reconstruction loss measures the accuracy of the reconstructed adjacency matrix compared to the original, while the regularization term encourages the latent representations to follow a Gaussian distribution, typically using the Kullback-Leibler (KL) divergence.

Note: In this study, VGAEs are categorized under the embedding-based models despite also being a GNN-based approach. This classification is due to the methodology used for clustering in our experiments, where the embeddings generated by VGAE are clustered using k-means, which is non-GNN-based.

D. GNN-based Approaches for Graph Clustering

GNNs have introduced new approaches to graph clustering, leveraging both node features and the structural attributes of graphs. A key component in these approaches is the use of specialized pooling layers [5], which aggregate node features, facilitating the process of cluster formation within graphs. Some of the most prominent pooling layers are the following:

1) MinCut Pooling: MinCut [2] computes soft cluster assignments as:

$$S = \text{softmax}(\text{MLP}(\tilde{X}, \Theta_{MLP})) \in \mathbb{R}^{N \times K} \tag{2}$$

where K is the number of clusters, $TildeX$ is the node feature matrix generated by the Message Passing (MP) [3] layers of the GNN and MLP$(\cdot)$ is a Multi-Layer Perceptron with trainable parameters Θ_{MLP}.

MinCut optimizes the following unsupervised loss function:

$$\mathcal{L}_{MC} = -\frac{\text{Tr}\left(S^T \tilde{A} S\right)}{\text{Tr}\left(S^T \tilde{D} S\right)} + \left\| \frac{S^T S}{\|S^T S\|_F} - \frac{I_K}{\sqrt{K}} \right\|_F, \tag{3}$$

where $\|\cdot\|$ is the Frobenius norm, $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ and $\tilde{D}$ is the degree matrix of $\tilde{A}$. The first term, $\mathcal{L}_c$ (MinCut Loss) minimizes the Local Quadratic Variation (LQV), while the second, $\mathcal{L}_o$ (Orthogonality Loss), is a balancing term. The computational complexity of MinCut is $\mathcal{O}(N^2K + NK^2)$, which is reduced to $\mathcal{O}(EK + NK^2)$ when using sparse operations.

2) DMoN Pooling: DMoN [16], similarly to MinCut, calculates the soft cluster assignments S ulitizing an MLP layer. To learn the cluster assignments, it optimizes the following modularity-based [13] loss function:

$$\mathcal{L}_{DMoN} = \frac{\text{Tr}\left(S^T \tilde{A} S\right)}{2E} - \frac{\sqrt{K}}{N} \left\| \sum_i {S_i}^T \right\|_F - 1, \tag{4}$$

where $\|\cdot\|$ is the Frobenius norm, $\tilde{A} = A - D^T D$ and D is the degree vector of A. As with MinCut, the computational complexity of DMoN is $\mathcal{O}(N^2K + NK^2)$, or $\mathcal{O}(EK + NK^2)$ when using sparse operations.

3) JustBalance Pooling: JustBalance (JB) [1] also calculates the soft cluster assignments S ulitizing an MLP layer, but offers a simplification in the calculation of the loss function, which consists only of a balancing term:

$$\mathcal{L}_{JB} = -\text{Tr}(\sqrt{S^T S}) \tag{5}$$

Because the LQV term is removed in the above loss function, the computational complexity of JustBalance is $\mathcal{O}(NK^2)$, which is lower than that of the other two pooling layers for most graphs.

E. Graph Matching Networks

Graph Matching Networks (GMNs) [10] are a specialized type of Graph Neural Networks that take a pair of graphs as an input and compute a similarity score between them. Unlike embedding models that independently map each graph to a vector, GMNs evaluate the similarity of graphs directly through joint computation.

A key feature of GMNs is the inclusion of a cross-graph matching vector in each propagation layer. This vector evaluates how well a node in one graph matches with nodes in another graph, taking into account both the aggregated messages on the edges and the cross-graph relationships. GMNs can be trained using either graph pairs or triplets. When using Euclidean similarity, the objective functions are the following:

- For pairwise training:

$$L_{\text{pair}} = \mathbb{E}_{(\mathrm{G}_1,\mathrm{G}_2,\mathrm{t})}\left[\max\left\{0, \gamma - t(1 - d(G_1, G_2)\right\}\right] \quad (6)$$

  where $t \in \{-1, 1\}$ is the label indicating the pair similarity (positive or negative), $d(G_1, G_2) = \|h_{G_1} - h_{G_2}\|^2$ is the Euclidean distance, and γ is a margin parameter.
- For triplet training:

$$L_{\text{triplet}} = \mathbb{E}\left[\max\left\{0, d(G_1, G_2) - d(G_1, G_3) + \gamma\right\}\right] \quad (7)$$

  where G_1, G_2, and G_3 are graph triplets with G_1 and G_2 being more similar than G_1 and G_3.

III. Methods

In this section, we discuss the the experimental setup, the tested methods and the implementation of the models that are used. In the first part of the experimental process,

A. Experimental Setup

We evaluate the clustering performance of the models in three citation datasets [**?**], which are presented in Table I, and select the number of clusters to be equal to the number of classes in the each dataset.

TABLE I: Datasets

Name	Nodes	Edges	Features	Classes
Cora	2708	10 556	1433	7
Citeseer	3327	9104	3703	6
PubMed	19 717	88 648	500	3

To assess the performance of the clustering methods, the following evaluation metrics are used:

1) Normalized Mutual Information (NMI):

$$\text{NMI}(U, V) = \frac{2 \times I(U; V)}{H(U) + H(V)}, \quad (8)$$

where $I(U; V)$ is the mutual information between the clustering results U and the true labels V, and $H(U)$ and $H(V)$ are the entropies of U and V respectively.

2) Accuracy (ACC) [1]:

$$\text{ACC} = \frac{1}{N}\sum_{i=1}^{N} \theta(y_i, h(\bar{s}_i)), \quad (9)$$

where $h(\cdot)$ maps the hard-cluster assignment $\bar{s}_i$ to the best matching class based on the Kuhn-Munkres algorithm, and θ is the Heaviside step function.

3) Adjusted Rand Index (ARI):

$$\text{ARI} = \frac{\text{RI} - \text{Expected RI}}{\text{Max RI} - \text{Expected RI}}, \quad (10)$$

where RI (Rand Index) measures the similarity between two clusters.

4) GMN Similarity Evaluation:

$$\text{SIM} = \text{GMN}(G_{\text{original}}, G_{\text{clustered}})$$

This metric measures the similarity score between the original graph G_{original} and the clustered graph $G_{\text{clustered}}$ as computed by the GMN.

B. Implementation of Clustering Methods

1) Embedding-based approaches: For the embedding-based clustering methods, we implemented Spectral Clustering, Node2Vec, and VGAE in a straightforward manner. Each method was applied to the graph data to generate the node embeddings, which were then clustered using k-means. This approach allowes for a direct comparison of the clustering outcomes based on different embedding techniques. It is important to note the Spectral Clustering and Node2Vec generate the node embeddings based only on the adjacency matrix, while VGAE generates the node embeddings by accounting both for the adjacency matrix and for the node features.

2) GNN-based approaches: In the GNN-based clustering methods, the models, through the MP and MLP layers, generate the soft cluster assignments S, by accounting for both the graph connectivity and the node features and, subsequently, the hard cluster assignments are computed as $\bar{s} = \text{argmax}(S)$. To make the comparison fair between MinCut, DMoN and JustBalance, the GNN architectures are configured to have the same capacity (number of layers and trainable parameters) and are trained for the same number of epochs.

C. GMN Integration

In the second phase of experimentation, a pretrained GMN is utilized in two distinct ways: as an auxiliary evaluation tool for the clustering models and as an integrated component in the clustering models' objective function.

1) GMN Pretraining: Following the methodology outlined in the original paper [10], the GMN is pretrained on the Graph Edit Distance problem. Graph edit distance between graphs G_1 and G_2 is defined as the minimum number of edit operations needed to transform G_1 to G_2. During the training, we generate training data by sampling random binomial graphs G_1 with n nodes and edge probability p, and then we create positive (G_2) or negative (G_3) examples by substituting k_p or k_n edges respectively, where $k_p < k_n$. Therefore, the model

	Cora				CiteSeer				PubMed			
	NMI	ACC	ARI	SIM	NMI	ACC	ARI	SIM	NMI	ACC	ARI	SIM
SC	0.041	0.291	−0.002	-	0.024	0.242	0.016	-	0.182	0.587	0.129	-
Node2vec	0.081	0.243	0.225	−0.231	0.06	0.197	0.02	−0.382	0.002	0.368	0.001	0.087
VGAE	0.535	0.712	0.523	0.665	0.262	0.437	0.136	0.558	0.284	0.665	0.271	0.591
MinCut	0.382	0.528	0.293	0.711	0.247	0.470	0.230	0.513	0.248	0.578	0.244	0.330
DMoN	0.292	0.419	0.179	−0.405	0.138	0.330	0.106	0.218	0.118	0.437	0.093	−0.408
JustBalance	0.361	0.479	0.266	0.253	0.162	0.394	0.112	0.259	0.202	0.570	0.160	0.518

TABLE II: Clustering performance metrics for each dataset

needs to predict a higher similarity score for the positive pair (G_1, G_2) compared to the negative pair (G_1, G_3).

Note: It is important to note that the limited available computational resources may have constrained the extent and depth of this training. This limitation is acknowledged and could impact the GMN's effectiveness in the experimental evaluations that follow.

2) Clustering Evaluation: During the validation phase of the clustering methods outlined in Section III-B, the pretrained GMN is used to compute the similarity between the original graph and the graph after clustering. This serves as a tool to evaluate how closely the clustering process retains the structural characteristics of the original graph.

3) Integration in Training: In the training process of the clustering models, the pretrained GMN was integrated into the loss function in two distinct ways:

- **Direct Integration**: From the beginning of training, the similarity score provided by the GMN is combined with the pooling loss. This approach aimed to directly influence the clustering process by incorporating graph structural information as part of the model's learning objective.
- **Fine-Tuning**: In this approach, the models are initially trained using only the pooling loss. After this initial training phase, the models are fine-tuned by adding the GMN-based loss to the existing pooling loss. This two-step process allows the models to first learn to cluster based on the primary method, and then refine the clustering by considering the graph's structural similarity as indicated by the GMN.

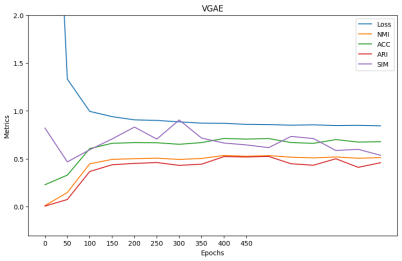
Both methods were explored to assess the impact of GMN-based loss on the evaluation metrics and the overall clustering performance.

IV. Results

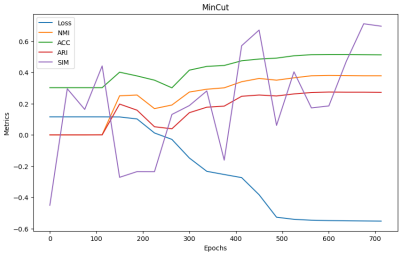
This section presents the outcomes of the experimental process, focusing on both the standalone performance of the clustering methods and the impact of integrating GMN similarity as an evaluation metric.

A. Performance of Clustering Methods

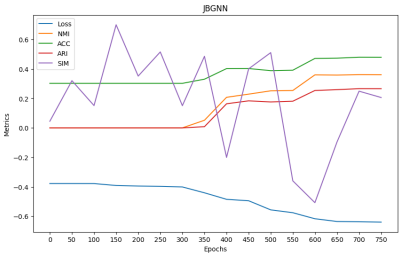
The model with the best overall performance was VGAE, which achieved the highest metrics across all datasets, by a significant margin. Along with MinCutPool, they also the only



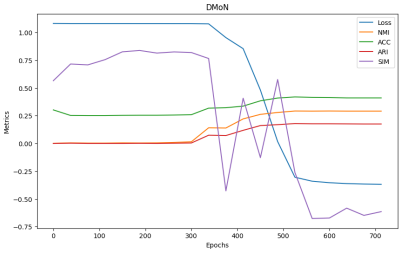
(a) VGAE



(b) MinCutPool



(c) JustBalance



(d) DMoN

Fig. 1: Graphics on Cora

models that had consistently high similarity scores across all datasets, as outlined in Figure 1. The only disadvantage of this method is that it is a two-step procedure, which in some cases is undesirable. The other two embeddings-based approached achieved worse scores in all metrics compared to the other models, which is expected, as they generate the embeddings based only on the graph connectivity.

Between the GNN-based methods, MinCut consistently achieved the best results across datasets, while DMoN achieved the worst. JustBalance's clustering performance was comparable to MinCut's in both "Cora" and "PubMed", despite the simpler objective function, an observation that agrees with the findings of the original paper [1], and, due to the lower computational complexity, achieved significantly faster training times.

An intriguing observation emerged from our analysis regarding the consistency of similarity scores across different models. Contrary to VGAE and MinCut, which displayed stable similarity scores, JustBalance and DMoN exhibited significant fluctuations in their similarity scores over epochs, as illustrated in Figure 1 and Table II. Notably, in the case of DMoN on the "Cora" and "PubMed" datasets, we observed a trend of decreasing similarity scores across epochs, often registering high negative values for most of the training period. This phenomenon was not observed with MinCut, which showed similarity trends similar to VGAE.

This inconsistency in similarity scores, particularly with DMoN and JustBalance, highlights that traditional metrics like NMI, ACC, and ARI, while closely related and displaying similar trends, do not necessarily correlate directly with changes in graph similarity. The divergence in trends between these traditional metrics and the GMN-based similarity scores suggests that an increase or decrease in NMI, ACC, and ARI does not automatically imply a corresponding increase or decrease in the structural similarity of the graphs.

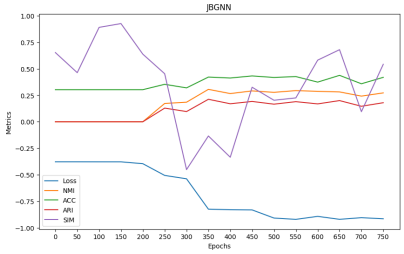
We address this phenomenon in the following section (IV-B) and explore methods to resolve it, by utilizing GMNs in the training process.

B. Impact of GMN

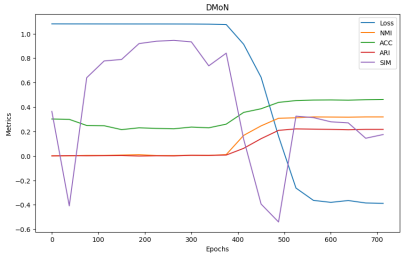
In this second part of the experimental evaluation, we focus on the "Cora" dataset and utilize the similarity metric provided by the GMN in order to improve the clustering performance of JustBalance and DMoN in the other metrics and limit the similarity inconsistencies we observed in the previous section (IV-A).

The similarity-based loss we use in this process is calculated as: $\mathcal{L}_{SIM} = l \cdot \frac{1-sim}{2}$, where sim is the similarity between the original and the clustered graph, as calculated by the GMN, and l is a hyperparameter that is used to scale the loss.

- **Direct integration**: Combining the similarity loss with the pooling loss from the beginning of the training has a positive impact on the performance of models, as seen in Figure 2. They are able to achieve comparable performance to the results from Table II in the traditional evaluation metrics (NMI, ACC and ARI) and the similarity, while still unstable, remains positive and increases over time.



(a) JustBalance



(b) DMoN

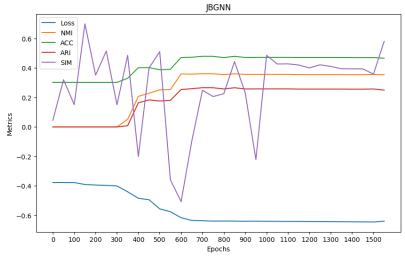
Fig. 2: Direct Integration

- **Fine Tuning**: The approach of initially training the models with only the pooling loss and subsequently integrating the similarity loss into the objective function yielded promising results. Specifically, in the case of JustPooling, this fine-tuning strategy led to an increase and stabilization in the similarity scores, without adversely affecting the performance as measured by other metrics. Conversely, for DMoN, there was an initial improvement in similarity scores, which became more stable towrds the end of the training process. Moreover, this fine-tuning approach also contributed to an improvement in the other evaluation metrics for DMoN.

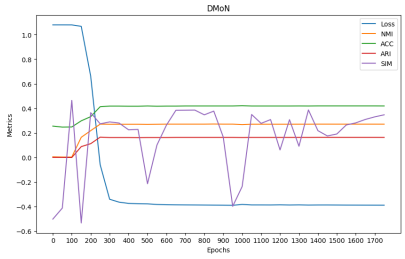
V. Conclusion

In this study, we investigated various Graph Neural Network (GNN)-based methods that leverage graph pooling layers and learned embeddings for graph clustering. Our comparative analysis revealed that these GNN-based approaches outperform traditional and non-GNN-based methods, demonstrating superior results across all evaluated metrics on a variety of citation datasets.

Additionally, we employed a pretrained Graph Matching Network (GMN) to assess the similarity between the original graphs and their clustered counterparts. This evaluation highlighted an interesting observation: the GMN-derived similarity metric does not always correlate directly with other metrics and shows considerable variation over different epochs. To address this, we integrated the GMN-based similarity measure into the objective function of the GNN-based models. This



(a) JustBalance



(b) DMoN

Fig. 3: Fine-Tuning

integration not only increased the similarity scores over time, enhancing their stability, but also did so without compromising the performance as measured by other metrics.

These findings underscore the potential of incorporating GMN-based assessments in GNN models for graph clustering. By doing so, we can improve the clustering process, ensuring not just high performance in traditional metrics, but also a closer structural alignment between the original and clustered graphs.

REFERENCES

[1] Filippo Maria Bianchi. Simplifying clustering with graph neural networks. *Proceedings of the Northern Lights Deep Learning Workshop*, 4, January 2023.

[2] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling, 2020.

[3] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering, 2017.

[4] Alessandro Generale, Till Blume, and Michael Cochez. Scaling r-gcn training with graph summarization. In *Companion Proceedings of the Web Conference 2022*, WWW '22. ACM, April 2022.

[5] Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, page 1–11, 2022.

[6] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks, 2016.

[7] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[8] Thomas N. Kipf and Max Welling. Variational graph auto-encoders, 2016.

[9] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[10] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects, 2019.

[11] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey, 2018.

[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[13] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

[14] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14. ACM, August 2014.

[16] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks, 2023.

[17] Ulrike von Luxburg. A tutorial on spectral clustering, 2007.

[18] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling, 2019.