Anchor-Driven Nyström for Deep Graph-Level Clustering

Jiaxin Wang[1], **Wenxuan Tu**[2], **Lingren Wang**[2], **Jieren Cheng**[2,3], **Yue Yang**[2]

[1]School of Cyberspace Security, Hainan University, Haikou, China
[2]School of Computer Science and Technology, Hainan University, Haikou, China
[3]Hainan Blockchain Technology Engineering Research Center, Haikou, China
jx_wang2046@163.com, twx@hainanu.edu.cn

Abstract

Graph-level clustering (GLC), which aims to group entire graphs according to their structural and attribute-based similarities, represents a fundamental yet challenging task in various practical applications. Existing GLC methods primarily fall into two main paradigms: 1) deep graph clustering approaches based on Graph Neural Networks (GNNs), and 2) kernel-based methods that utilize predefined kernels to perform fine-grained structural comparison for clustering. However, GNN-based methods typically learn graph-level representations by aggregating node embeddings through pooling operations, which inevitably leads to substantial information loss and suboptimal clustering performance. In contrast, kernel methods, despite their theoretical discriminability, suffer from prohibitive computational costs that hinder their scalability to large-scale settings. To solve these issues, we propose a novel graph learning framework named **A**nchor-driven **N**yström for Deep **G**raph-Level **C**lustering (**ANGC**), which computes graph similarity via kernel methods while retaining the scalability of GNNs. Specifically, we first employ GNNs to encode individual graphs into sets of node embeddings. Rather than relying on pooling operations, we compute graph similarities in a kernel space constructed from these embeddings. To enhance both scalability and expressive power, we introduce learnable graph Nyström anchors, which support end-to-end optimization and significantly accelerate kernel computations. To further improve the discriminative capability of these anchors, we propose the concept of anchor response discrepancy, that is, the variation in a given anchor's responses across different samples. By maximizing this discrepancy, the anchors are encouraged to strengthen inter-graph distinctions for better clustering. Extensive experiments demonstrate the effectiveness and superiority of ANGC over existing state-of-the-art methods.

Code — https://github.com/JXWANG-GRAPH/ANGC

Introduction

Graph-level clustering (GLC) is a fundamental unsupervised learning task that aims to group entire graphs such that those within the same cluster exhibit high structural and semantic similarity, while graphs in different clusters are dissimilar (Chen et al. 2025). In recent years, Graph Neural Networks (GNNs) have become the dominant paradigm for graph learning (Tu et al. 2022, 2024b,a; Hu et al. 2025b; Liu et al. 2025b), owing to their scalability and strong ability to capture task-specific graph semantics.

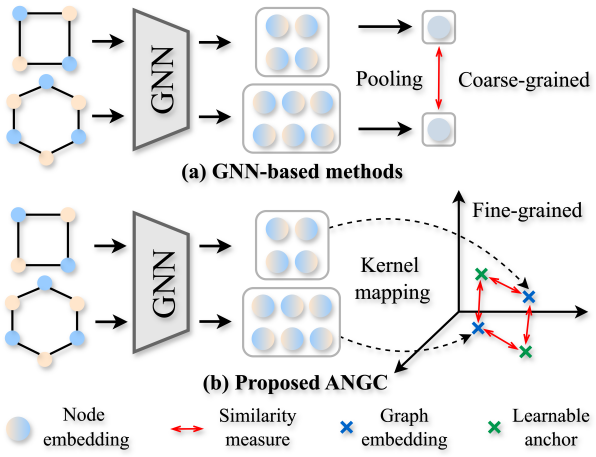

Figure 1: Comparison of learning paradigms between existing GNN-based deep clustering methods and the proposed ANGC. (a) Although graphs may exhibit subtle structural differences, the coarse representations generated by GNN pooling often lack sufficient discriminative power. (b) In contrast, ANGC measures graph similarity in a kernel space without relying on pooling, enabling more fine-grained and expressive similarity comparisons. Furthermore, by introducing learnable anchors, ANGC supports end-to-end training while maintaining low computational overhead.

According to the graph similarity metric, the state-of-the-art graph-level clustering methods can generally be categorized into two main types: deep graph clustering based on GNNs and graph kernel-based clustering approaches (Bai et al. 2024a; Li et al. 2024). The GNN-based GLC methods typically begin with encoding structural information into node embeddings by the message-passing mechanism (Tu et al. 2024c; Bicciato et al. 2024; Liang et al. 2024; Deng et al. 2025). These node embeddings are then aggregated (e.g., via sum or mean pooling) to obtain coarse-grained graph-level representations, which are used for clustering by measuring graph similarity (You et al. 2021; Cai et al.

Corresponding authors: Jieren Cheng, Wenxuan Tu

Method	Scalability	Discrimination	Expressiveness	Flexibility
Kernel-based methods	✗	✔	✗	✔
GNN-based methods	✔	✗	✔	✗
ANGC (Ours)	✔	✔	✔	✔

Table 1: Comparison with related work. "Scalability" indicates that the method could be easily extended to large-scale scenarios. "Discrimination" refers to the extent to which the learned representations exhibit strong discriminative capability. "Expressive power" denotes the capacity to learn high-quality and robust representations tailored to specific tasks. "Flexibility" reflects the ability to accommodate diverse graph similarity measures.

2022b; Bai et al. 2024b; Zhang et al. 2025a), as illustrated in Fig. 1(a). Despite their success in scalability and expressive power, pooling operations often compromise the discriminative quality of graph representations by mapping structurally distinct graphs to similar embeddings, due to substantial information loss. This limitation hinders the resulting graph-level representations from accurately capturing inter-graph distinctions for clustering. In contrast, kernel methods employ predefined kernels to explicitly capture both structural and attribute similarities, enabling fine-grained graph comparisons without relying on pooling (Shervashidze et al. 2011; Zaripova et al. 2023; Cosmo et al. 2024; Wang, Tu, and Cheng 2025). Moreover, their flexibility in defining diverse kernels allows them to characterize graph similarity from multiple complementary perspectives. However, the high computational complexity of kernel methods severely limits their scalability to large-scale datasets (Wang et al. 2023; Zhang et al. 2025b). Additionally, their inability to be jointly optimized with the clustering objective constrains their capacity to learn representations that are well-aligned with downstream clustering tasks (Qian et al. 2024). These challenges motivate the development of novel frameworks that seamlessly integrate the fine-grained similarity modeling of kernel methods with the scalability and end-to-end representation learning advantages of GNNs.

An intuitive solution is to develop a framework that effectively reconciles the fine-grained graph comparison capabilities of kernel methods with the scalability and representation power of GNNs. Within this framework, a GNN is employed to encode the structural and feature information of graphs, followed by the application of kernel methods to perform graph similarity measurement based on these learned representations. To achieve this, two key challenges are supposed to be addressed: 1) the kernel-driven graph similarity computation must be computationally efficient, ideally with linear complexity; and 2) how to integrate graph similarity measurement with GNNs into a unified optimization framework. To tackle the first challenge, we draw inspiration from the Nyström method (Williams and Seeger 2000), which approximates kernel computations by selecting a subset of anchors to reduce computational overhead significantly. For the second challenge, we parameterize these anchors as learnable entities, enabling joint optimization of kernel computation and GNN representation learning. This design allows the entire framework to be trained end-to-end, effectively unifying the strengths of both GNNs and kernel methods while aligning the learned representations with the clustering objective.

In this work, we design a simple yet effective kernel-based deep graph-level clustering framework, termed **A**nchor-Driven **N**yström for Deep **G**raph-Level **C**lustering (**ANGC**). Specifically, we first employ GNNs to encode the structural and attribute features of each graph into node embeddings. These node embeddings are then used to compute graph similarity via kernel methods. To make this process both end-to-end trainable and computationally efficient, we design the parameterized graph Nyström anchors that serve as representative proxies in the similarity computation. Moreover, to enhance the discriminative capacity of these anchors, we propose response difference regularization, which encourages the model to learn more informative anchor representations by maximizing the variability in responses of different graphs to the same anchor. In the end, we seamlessly integrate graph similarity measurement with GNNs representation learning into a unified framework. On one hand, by leveraging the fine-grained graph comparison of kernel methods, we can more accurately measure graph similarity while avoiding the information loss caused by pooling. Moreover, different base kernels can be flexibly employed to capture various structural patterns. On the other hand, learnable graph Nyström anchors reduce the kernel computation to linear complexity. The end-to-end design enables the clustering objective to directly guide the GNNs in learning more expressive representations. To highlight the advantages of our method over existing approaches, we conduct a comparative analysis of key characteristics, including scalability, discrimination, expressiveness, and flexibility (see Table 1). We summarize the contributions of this work:

- We propose ANGC, an end-to-end clustering framework that seamlessly integrates the fine-grained comparison strengths of kernel methods with the scalability and representation learning power of GNNs within a unified optimization architecture.
- We propose learnable graph Nyström anchors, which enable joint optimization of kernel-based graph similarity measurement and the clustering objective in an end-to-end manner. Additionally, we introduce a novel anchor response discrepancy regularization to enhance the discriminative power of these anchors, thereby improving clustering performance.
- Extensive experiments demonstrate that ANGC achieves superior performance and computational efficiency compared to state-of-the-art competitors.

Related Work

GNN-based GLC methods typically leverage graph neural networks to encode structural information into node representations, which capture local substructure patterns (Hu et al. 2025a; Tu et al. 2025; Liu et al. 2025a). These node representations are then aggregated into graph-level embeddings via a pooling operation, which are subsequently used for clustering. For example, DCGLC (Cai et al. 2024) integrates subspace clustering techniques and performs clustering in both subspace (Cai et al. 2022a) and Euclidean space after pooling, aiming to enhance performance through multiple clustering criteria. GPC (Chen et al. 2025) leverages pretrained GNNs with prompt-based finetuning to adapt graph-level clustering to diverse data distributions. Despite their successes, these approaches rely heavily on graph-level representations derived from GNNs, which may cause significant information loss during the compression process and thus limit clustering performance. Although various graph pooling methods have been proposed, such as the clustering-based DiffPool (Ying et al. 2018), the sorting-based SortPool (Zhang et al. 2018), and the attention-based SAGPool (Lee, Lee, and Kang 2019), they enhance hierarchical graph representations only to a limited extent. Due to the inevitable loss of nodes or edges during pooling, these approaches often preserve limited structural information.

In contrast to GNN-based methods, graph kernel techniques are the predominant tools for graph learning before the emergence of GNNs. These methods rely on predefined substructure (e.g., path, subtree, or subgraph) to perform fine-grained comparisons of structural patterns (Borgwardt and Kriegel 2005; Shervashidze et al. 2011; Cosmo et al. 2024; Zhang et al. 2024), often resulting in stronger expressive power. While graph kernels enhance representational power, they also incur substantial computational costs (Liu et al. 2025c). To reduce this complexity, prior work introduced the Nyström method (Williams and Seeger 2000), which approximates the full kernel matrix by randomly selecting a subset of samples as anchors, thereby reducing the computational cost from quadratic to linear. However, despite this efficiency gain, their non-end-to-end nature prevents joint optimization with downstream tasks, limiting their capacity to learn task-specific representations for clustering. In this paper, we introduce graph Nyström anchors, enabling the seamless integration of kernel methods and clustering objectives into an end-to-end framework.

Method

In this part, we begin with an illustration of the basic notations and definitions, and then present the proposed ANGC method in detail.

Notations and Definitions

A graph $G = \{\mathbf{X}, \mathbf{A}\}$ with n nodes is represented by two matrices: the adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$, where $\mathbf{A}_{ij} = 1$ indicates an edge between node i and j; and the node feature matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, where the i-th row $\mathbf{x}_i \in \mathbb{R}^d$ represents the feature vector of node i. A kernel function $k(\cdot,\cdot)$ measures the similarity between two objects and implicitly defines a mapping $\psi(\cdot)$ into a high-dimensional Hilbert space. Given a set of inputs $\{x_1,\dots,x_n\}$, the kernel matrix K collects similarities with entries $K_{ij} = k(x_i, x_j)$.

Definition 1 (**Nyström Feature Mapping**). *Given a positive semi-definite kernel function $k(x,x')$, the Nyström method approximates the implicit mapping $\psi(x)$ induced by the kernel using a set of M anchor points $\{x_1,\dots,x_M\} \subset \mathcal{X}$. The Nyström feature mapping is defined as:*

$$\psi(x) = K_{x,\mathcal{A}} K_{\mathcal{A},\mathcal{A}}^{-1/2}, \tag{1}$$

where $K_{x,\mathcal{A}} = [k(x,x_1),\dots,k(x,x_M)] \in \mathbb{R}^{1\times M}$ denotes the kernel similarities between x and the anchors $\mathcal{A} = \{x_1,\dots,x_M\}$, and $K_{\mathcal{A},\mathcal{A}} \in \mathbb{R}^{M\times M}$ is the kernel matrix over the anchors. This yields a finite-dimensional, explicit approximately kernelized representation of x that preserves the similarity structure encoded by k. In this paper, $\psi(x) \in \mathbb{R}^M$ denotes the vector representation of x.

Anchor-Driven Nyström Method

Graph Pre-Encoding Before training ANGC, we first generate an augmented graph $\tilde{G} = \{\tilde{\mathbf{X}}, \tilde{\mathbf{A}}\}$ from the original graph G, following the augmentation strategy proposed in GraphCL (You et al. 2020). Both the original graph and its augmented view are then encoded using a shared graph encoder f_E, which is implemented with GIN (Xu et al. 2019). The entire pre-encoding process can be formalized as:

$$\mathbf{H}_G = f_E(\mathbf{X}, \mathbf{A}), \quad \mathbf{H}_{\tilde{G}} = f_E(\tilde{\mathbf{X}}, \tilde{\mathbf{A}}), \tag{2}$$

where $\mathbf{H}_G, \mathbf{H}_{\tilde{G}} \in \mathbb{R}^{n\times d}$ and n is the number of nodes of graph G and d is the embedding dimension. These representations are then leveraged to measure the similarity between graphs in the subsequent modules. Prior work (Xu et al. 2019) has shown that the node representations obtained from an L-layer GNNs implicitly capture the L-hop subtree structures rooted at each node, which are provably equivalent to the features extracted by the 1-WL (Weisfeiler–Lehman) test (Shervashidze et al. 2011). These representations are then leveraged to measure the similarity between graphs in the subsequent modules.

Deep Nystrom Feature Mapping To reduce the computational complexity of kernel computing, we employ the Nyström method (Williams and Seeger 2000) to accelerate kernel computing. This enables costly kernel methods to scale to large datasets while maintaining strong performance. However, traditional Nyström approaches rely heavily on the choice of anchor points, and the quality of the approximation is highly sensitive to their representativeness. Moreover, their non-end-to-end nature hinders seamless integration with deep learning models and joint optimization. To address these limitations, we propose a novel formulation of learnable anchors, i.e., parameterized anchors that can be optimized jointly with the learning objective.

Definition 2 (**Graph Nyström Anchors**). *Let $\mathcal{A} = \{\mathbf{Z}_1, \mathbf{Z}_2, \dots, \mathbf{Z}_M\}$ be a set of M **parameterized** graph anchors, where each $\mathbf{Z}_m \in \mathbb{R}^{n\times d}$ corresponds to the node embeddings matrix of the m-th anchor graph. Here, $\mathbf{Z}_m$ is a trainable parameter, meaning that its entries are optimized*

during the learning process. The integer n denotes the number of nodes in the anchor graph, and d is the dimensionality of the node representations.

The anchor set $\mathcal{A}$ can be jointly optimized with the training objective, enabling the anchors to serve as representative graph prototypes that facilitate graph-level representation learning and kernel approximation. Based on these learnable anchors, we then compute the Nyström embedding of a graph G in the anchor space $\mathcal{A}$ according to Eq. (1):

$$\psi(G) = K_{G,\mathcal{A}} K_{\mathcal{A},\mathcal{A}}^{-1/2}. \tag{3}$$

This embedding is then used as the graph representation for clustering via K-means. For this base kernel $k(\cdot,\cdot)$ computation in Eq. (3), we first use a distance metric to quantify the difference between graphs G_i and G_j based on their encoded node representations $\mathbf{H}_{G_i}$, $\mathbf{H}_{G_j}$, denoted as $d(G_i, G_j)$, and then apply a Gaussian kernel to convert the distance into a similarity score, that is:

$$k(G_i, G_j) = \exp\left(-\lambda d(G_i, G_j)^2\right), \tag{4}$$

where λ is a hyperparameter that controls how strongly the distance $d(G_i, G_j)$ between graphs affects their similarity. In this paper, we use Maximum Mean Discrepancy (MMD) (Borgwardt et al. 2006) as the metric to measure the graph distance $d(\cdot,\cdot)$.

Through the learnable graph Nyström anchors, we explicitly map each graph into the kernel space induced by a set of anchor embeddings, thereby approximating the original kernel features. Most importantly, the parameterized design enables these embeddings to be directly used in downstream clustering tasks, allowing the representations to be optimized in an end-to-end manner.

Response Discrepancy Regularization While parameterized anchors provide flexibility through end-to-end learning, their discriminative ability is not guaranteed without explicit guidance. In practice, anchors with poor discrimination may fail to effectively represent diverse graph structures, limiting the quality of the resulting embeddings. To enhance the discriminative power of the anchors, we introduce a dedicated optimization objective that encourages each anchor to respond distinctly to different graphs.

Definition 3 (Anchor Response Discrepancy). *Let $K_{XZ} \in \mathbb{R}^{N\times M}$ denote the kernel matrix between N graphs and M anchors, where $[K_{XZ}]_{i,j} = k(x_i, z_j)$ denotes the similarity between graph i and anchor j. The response discrepancy of the j-th anchor is defined as:*

$$r_j = \sum_{1\le i<i'\le n} \left(K_{XZ}[i,j] - K_{XZ}[i',j]\right)^2. \tag{5}$$

In practice, the response discrepancy of all anchors can be computed in closed form using matrix operations:

$$\mathbf{r} = N \cdot \mathrm{diag}(K_{XZ}^\top K_{XZ}) - \left(K_{XZ}^\top \mathbf{1}_N\right)^{\circ 2}, \tag{6}$$

where $\mathbf{r} \in \mathbb{R}^M$ is the vector of response discrepancies for all anchors, $\mathbf{1}_N \in \mathbb{R}^N$ is an all-one column vector, $\mathrm{diag}(\cdot)$ extracts the diagonal of a matrix and $(\cdot)^{\circ 2}$ denotes element-wise square. The response discrepancy of the j-th anchor, denoted as r_j, quantifies the variability in responses of different samples to this anchor. A high value of r_j indicates that the anchor produces diverse responses across the sample set, suggesting a strong discriminative capacity. In contrast, a low response discrepancy implies that the anchor yields similar responses for all samples, indicating limited discriminative power.

Based on anchor response discrepancy, we propose response difference regularization, which maximizes the total response discrepancy across all anchors:

$$\mathcal{L}_R = -\sum_{i=1}^{M} r_i. \tag{7}$$

This target loss promotes diversity and separation among anchors, improving their capacity to capture meaningful structural variations in the data.

Graph Encoder Optimization We now specify the optimization for the graph encoder f_E. For simplicity, we design our optimization strategy based on the contrastive learning paradigm (You et al. 2020).

Specifically, we obtain the Nyström embeddings $\psi(G)$ and $\psi(\tilde{G})$ of the original graph G and the augmented graph $\tilde{G}$ according to Eq. (3). These embeddings are then used to compute the contrastive loss. The NT-Xent loss (Sohn 2016) is adopted in this paper as the contrastive loss, designed to enhance the similarity between positive pairs while pushing apart negative pairs. Given a positive pair of graph embeddings $\left(\psi(G), \psi(\tilde{G})\right)$, the loss is defined as:

$$\ell_G = -\log \frac{\exp\left(\mathrm{sim}\left(\psi(G), \psi(\tilde{G})\right)/\tau\right)}{\sum\limits_{G'\in\mathcal{B}, G'\neq G} \exp\left(\mathrm{sim}\left(\psi(G), \psi(\tilde{G'})\right)/\tau\right)}, \tag{8}$$

where $\mathrm{sim}(\cdot,\cdot)$ denotes the cosine similarity between two embeddings, τ is a temperature parameter, and $\mathcal{B}$ is the set of all graphs in the batch. The total contrastive loss is then computed by averaging over all graphs in the mini-batch:

$$\mathcal{L}_K = \frac{1}{|\mathcal{B}|} \sum_{G\in\mathcal{B}} \ell(G). \tag{9}$$

Cluster Refinement We have respectively designed optimization strategies for the anchors and the graph encoder. However, these strategies are not inherently designed for deep clustering. To provide more reliable guidance for clustering, we follow previous work (Tu et al. 2021) and adopt a self-supervised approach to enhance the clustering process. Specifically, we use the Student's t-distribution to generate soft assignments for each sample:

$$q_{ij} = \frac{(1 + \|\psi(G_i) - \mu_j\|^2/t)^{-\frac{t+1}{2}}}{\sum_{j'}(1 + \|\psi(G_i) - \mu_{j'}\|^2/t)^{-\frac{t+1}{2}}}, \tag{10}$$

where q_{ij} denotes the probability that graph G_i belongs to cluster j, and μ_j is the K-means cluster center for cluster j.

Algorithm 1: The learning procedure of ANGC

Input: Graph datasets $\mathcal{G} = \{G_1, ..., G_N\}$; number of epochs E; number of Nyström anchors M and nodes of per anchor graph n;
Output: The clustering results.

```
Initialize graph encoder f_E and graph Nyström anchors.
for e = 1 to E do
    Generate augmented graphs G̃ for G ∈ G.
    Obtain graph embeddings H_G, H_G̃ by Eq. (2).
    Obtain Nyström embeddings ψ(G), ψ(G̃) by Eq. (3).
    Compute total loss by Eqs. (7), (9), (12), (13).
    Back-propagation and update graph encoder f_E and
    graph Nyström anchors.
end for
Obtain the clustering results via K-means on ψ(G).
return The clustering results.
```

Subsequently, to increase the confidence of cluster assignments, we derive a target distribution based on the soft assignments:

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{j'} (q_{ij'}^2 / \sum_i q_{ij'})}. \tag{11}$$

We define $Q = [q_{ij}]$ as the soft assignment matrix and $P = [p_{ij}]$ as the target distribution for all samples. The cluster refinement loss is then defined as:

$$\mathcal{L}_C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{12}$$

Training Strategy The graph encoder f_E and graph Nyström anchors are jointly optimized by minimizing the following total objective function:

$$\mathcal{L} = \mathcal{L}_K + \beta \mathcal{L}_R + \gamma \mathcal{L}_C, \tag{13}$$

where β and γ are trade-off parameters that control the contributions of $\mathcal{L}_R$ and $\mathcal{L}_C$ to the total loss. The detailed training procedure is provided in Algorithm 1.

Complexity Analysis

The time complexity of ANGC primarily consists of two parts: the Nyström feature mapping and the graph encoder f_E. Given a batch of N graphs and M anchors, computing the kernel matrix $K_{\mathcal{A},\mathcal{A}}$ requires $\mathcal{O}(M^2 C_k)$, where C_k denotes the cost of the base kernel $k(\cdot,\cdot)$. Computing $K_{G,\mathcal{A}}$, the similarities between all samples and anchors, takes $\mathcal{O}(NMC_k)$, resulting in a total cost of $\mathcal{O}((NM + M^2)C_k)$ for the Nyström feature mapping. The time complexity of the encoder is $\mathcal{O}(L(E + nD_{\text{in}}D_{\text{out}}))$, where L is the number of GIN layers, E and n represent the numbers of edges and nodes, and D_{in}, D_{out} denote the input and output dimensions, respectively. Therefore, the overall time complexity of ANGC is $\mathcal{O}((NM + M^2)C_k + NL(E + nD_{\text{in}}D_{\text{out}}))$. Since M is typically small (set to 15–30 in our experiments), the total complexity grows nearly linearly with the number of graphs and the number of nodes and edges.

Dataset	#Graphs	#Avg. nodes	#Avg. edges
MUTAG	188	17.93	19.79
PTC-MR	344	14.29	14.69
BZR	405	35.75	38.36
NCI1	4110	41.22	43.45
DD	1178	284.32	715.66
ENZYMES	600	32.60	62.14
COLLAB	5000	74.49	2457.78
IMDB-BINARY	1000	19.77	96.53
REDDIT-M-12K	11929	391.41	456.89

Table 2: Dataset statistics.

Experiments

In this section, we first evaluate the performance of ANGC on graph clustering datasets. Next, we study the runtime behaviour of the proposed methods on large-scale datasets. Finally, we conduct ablation experiments and hyperparameter analysis on ANGC.

Experimental Setup

Datasets We evaluate the proposed framework on nine datasets from different domains and of varying scales, all included in the TUDatasets collection (Morris et al. 2020). These datasets are categorized as follows: small molecule datasets (MUTAG, PTC-MR, BZR, NCI1), bioinformatics datasets (DD, ENZYMES), and social network datasets (COLLAB, IMDB-BINARY, REDDIT-M-12K). The dataset statistics is in Table 2.

Baseline Methods We compare the proposed method with several state-of-the-art baselines, which can categorized into three main types: 1) Graph Kernels: SP (Borgwardt and Kriegel 2005), RW (Vishwanathan et al. 2010), WL-OA (Kriege, Giscard, and Wilson 2016), 2) Unsupervised Graph Learning: InfoGraph (Sun et al. 2019), GraphCL (You et al. 2020), JOAO (You et al. 2021), 3) Deep Graph-Level Clustering: GLCC (Ju et al. 2023), DCGLC (Cai et al. 2024), GPC (Chen et al. 2025). For unsupervised graph learning baselines, we employ K-means to obtain the clustering results.

Implementation Details The reported results of ANGC and all compared methods were conducted on the same device and under identical settings for fair comparison. We employ a 5-layer GIN as the graph encoder, with an output dimension of 32. The batch size is fixed at 128, and the number of training epochs is set to 200 for all datasets. We follow the graph augmentation strategy used in GraphCL (You et al. 2020). By default, the number of learnable anchors is set to 20, and each anchor corresponds to 20 nodes. The hyperparameter λ is chosen from the values $[0.001, 0.01, 0.1, 1]$. Moreover, the learning rate, dropout rate, and weight decay are set to $1e{-}3$, 0.3, and $5e{-}4$, respectively, by default. We employ clustering accuracy (ACC), normalized mutual information (NMI), and adjusted Rand index (ARI) as evaluation metrics, presenting the mean and standard deviation values of these metrics across 10 independent runs per dataset.

Dataset	Metric	SP	RW	WL	InfoGraph	GraphCL	JOAO	GLCC	DCGLC	GPC	ANGC
MUTAG	ACC	72.87±0.00	77.65±0.00	73.40±0.00	77.95±1.41	77.07±1.21	79.20±0.72	71.99±3.08	81.42±1.68	79.53±3.78	**85.18±3.53**
	NMI	10.24±0.00	30.81±0.00	14.50±0.00	35.22±3.47	35.69±2.83	36.32±3.03	13.18±6.93	37.58±3.18	37.01±3.96	**39.86±3.58**
	ARI	15.95±0.00	30.26±0.00	21.20±0.00	30.95±3.03	28.99±2.65	33.74±1.65	16.89±8.28	42.87±3.40	43.65±4.33	**45.61±2.58**
PTC-MR	ACC	56.69±0.00	56.98±0.00	52.91±0.00	54.79±0.68	54.33±0.76	56.39±0.18	56.10±3.29	**59.98±1.11**	55.69±1.59	58.26±1.21
	NMI	1.04±0.00	0.63±0.00	0.23±0.00	0.49±0.35	1.15±0.55	0.53±0.21	1.21±0.63	2.60±0.89	1.24±0.66	**2.65±0.35**
	ARI	0.50±0.00	1.25±0.00	0.05±0.00	0.28±0.21	0.16±0.029	0.41±0.01	1.38±1.15	**3.66±0.92**	0.81±0.22	2.73±0.50
BZR	ACC	79.51±0.00	64.69±0.00	75.56±0.00	63.62±2.41	71.43±4.09	72.64±4.26	63.62±9.79	81.73±0.00	74.24±4.38	**83.07±0.73**
	NMI	4.13±0.00	0.00±0.00	0.50±0.00	1.59±0.95	1.04±0.77	1.37±1.14	1.18±0.60	13.57±0.00	4.59±0.75	**14.78±0.47**
	ARI	3.97±0.00	-0.15±0.00	3.76±0.00	2.39±1.44	3.07±1.03	4.01±3.39	1.12±0.97	18.27±0.00	2.32±0.86	**18.75±0.33**
NCI1	ACC	50.10±0.00		50.05±0.00	55.21±0.41	54.82±1.13	51.55±2.23	56.31±1.50	56.94±1.27	59.18±2.85	**61.48±1.91**
	NMI	0.10±0.00	OOT	0.00±0.00	1.34±0.55	0.65±0.21	0.84±0.18	1.32±0.41	1.07±0.13	2.57±0.38	**2.67±0.14**
	ARI	0.00±0.00		0.00±0.00	0.94±0.39	1.08±0.14	0.51±0.22	1.01±0.35	1.31±0.42	3.15±1.32	**3.32±0.90**
DD	ACC	58.83±0.00		58.57±0.00	58.57±0.04	58.02±0.04	57.95±0.04	60.70±0.00	69.52±2.46	61.35±1.16	**75.27±4.18**
	NMI	0.43±0.00	OOM	0.13±0.00	0.64±0.00	0.67±0.00	1.49±0.00	2.40±0.00	13.77±1.62	7.69±0.79	**20.64±2.21**
	ARI	0.14±0.00		-0.05±0.00	-0.02±0.08	-0.33±0.03	-0.37±0.00	2.30±0.00	21.25±1.08	14.31±1.85	**23.53±2.34**
ENZYMES	ACC	22.00±0.00	17.00±0.00	21.00±0.00	22.06±0.98	21.50±0.22	21.66±0.37	21.82±0.54	22.10±0.61	25.95±0.92	**27.82±1.42**
	NMI	2.57±0.00	0.66±0.00	3.09±0.00	2.40±0.45	1.25±0.12	1.60±0.01	1.59±0.23	1.27±0.38	**6.27±0.75**	5.52±2.04
	ARI	1.69±0.00	0.25±0.00	-0.40±0.00	1.25±0.52	0.90±0.09	0.94±0.02	1.02±0.33	1.07±0.29	2.30±0.58	**2.41±0.41**
COLLAB	ACC	47.38±0.00		53.52±0.00	59.35±2.41	58.54±1.75	57.28±2.03	58.85±2.43	59.64±1.83	59.58±2.18	**64.21±2.73**
	NMI	16.57±0.00	OOT	1.85±0.00	13.85±0.78	17.01±0.95	18.55±1.14	17.52±1.35	19.71±1.42	22.89±2.23	**26.49±1.56**
	ARI	13.95±0.00		0.44±0.00	6.72±0.30	10.28±0.88	11.19±1.21	10.93±1.16	11.62±1.35	12.38±0.86	**17.31±1.85**
IMDB-BINARY	ACC	53.95±1.15	51.30±0.00	51.24±0.53	54.79±0.84	54.66±0.13	58.20±1.84	66.50±0.00	66.50±2.37	59.57±2.85	**70.83±2.31**
	NMI	6.52±1.69	0.16±0.00	0.69±0.79	4.77±0.16	5.16±0.19	6.76±1.19	8.10±0.00	9.16±2.09	7.32±1.84	**13.78±1.25**
	ARI	0.66±0.36	0.03±0.00	0.06±0.05	0.92±0.38	0.83±0.02	2.77±1.13	10.60±0.00	11.03±2.72	9.27±1.96	**15.39±1.64**
REDDIT-M-12K	ACC			18.91±0.00	20.36±0.79	18.97±1.11	18.35±0.75	22.60±0.00	21.69±0.05	18.46±0.58	**25.14±0.69**
	NMI	OOT	OOT	8.14±0.00	4.02±0.35	9.82±1.35	1.65±0.08	10.51±0.46	10.40±1.58	2.18±0.31	**15.57±0.62**
	ARI			4.85±0.00	0.87±0.18	2.65±0.32	0.48±0.00	4.99±0.23	3.18±0.43	4.04±0.28	**8.66±1.08**

Table 3: Clustering performance (means ± std) on nine graph datasets (%). "OOM" indicates the out-of-memory failure. "OOT" means cannot completed within 24 hours. The best and runner-up results are highlighted in **bold** and underline, respectively.

Dataset	GraphCL	DCGLC	GPC	ANGC
MUTAG	3.14	2.37	2.84	4.27
BZR	1.78	3.80	2.26	11.30
NCI1	17.34	69.42	49.15	77.10
COLLAB	187.24	245.28	135.83	218.94
REDDIT-M-12K	532.95	1069.05	872.41	559.14

Table 4: One-epoch training runtime (in seconds) of the evaluated baseline approaches.

Performance Comparison

Table 3 reports the experimental results on nine datasets covering diverse domains and scale, where ANGC is compared against nine state-of-the-art baselines. From the results, we draw the following key observations. First, across 9 datasets, ANGC achieves the highest accuracy on 8 of them, demonstrating its strong ability to learn graph representations that are well-suited for clustering. In particular, compared to state-of-the-art baselines such as GLCC, DCGLC and GPC, ANGC achieves clear and stable improvements across molecule, bioinformatics, and social network datasets. For example, on the datasets DD, IMDB-BINARY, and COLLAB, ANGC surpasses the closest competitor by 5.75%, 4.57%, and 4.33% in terms of ACC. Similarly, on the social network dataset COLLAB, IMDB-BINARY, and REDDIT-M-12K, ANGC outperforms the second-best method across all metrics. These results validate the effectiveness of incorporating kernel-based similarity modeling and the design choice to avoid information-lossy pooling operations in ANGC. Moreover, we observe that ANGC maintains strong performance on large-scale datasets, demonstrating its scalability and robustness. In contrast, several kernel-based baselines (e.g., SP, RW) fail to complete within the 24 hours time limit on datasets such as COLLAB and REDDIT-M-12K, due to their high computational complexity in computing pairwise kernel matrices. This highlights a major advantage of ANGC—its ability to scale to large graphs without sacrificing effectiveness.

Runtime Comparison To evaluate the scalability of our proposed ANGC compared to existing GNNs, we conducted experiments on large-scale datasets including NCI1, COLLAB, and REDDIT-M-12K, as well as on small-scale datasets MUTAG and BZR. The encoder was configured with the same architecture across all models, consisting of 5 layers with a hidden dimension of 32, 20 anchors, and 20 nodes per anchor graph. All experiments were conducted on a system equipped with an 8-core Intel Xeon E5-2667 v4 @ 3.2GHz CPU, 256 GB of RAM, and an NVIDIA RTX 3090 GPU. Table 4 reports the runtime comparison across different methods. From the results, we observe that on small-

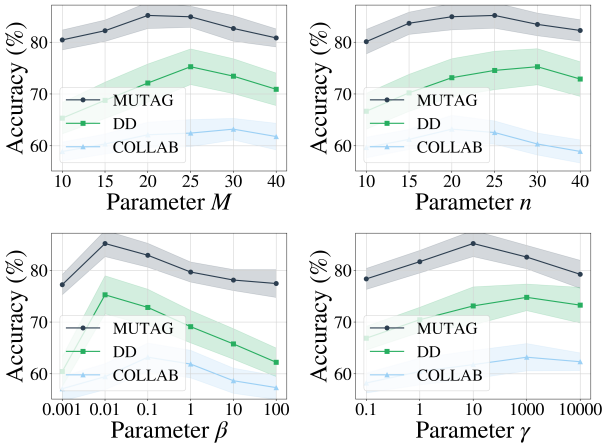

Figure 2: Hyperparameter sensitivity of ANGC on three benchmark datasets with respect to four key hyperparameters: number of anchors M, nodes per anchor n, β and γ.

scale datasets, ANGC incurs slightly higher runtime than other baselines, as the computation cost introduced by kernel methods is relatively more significant in this setting. However, as the dataset size increases, this kernel-induced cost is no longer dominant, and ANGC achieves substantially lower runtime than most methods, as demonstrated on REDDIT-M-12K. This also demonstrates that, despite incorporating kernel methods, our approach remains highly competitive with existing GLC methods.

Ablation Study

To examine the effect of learnable anchors and the anchor response discrepancy regularization, we evaluate two ANGC variants on five benchmark datasets: ANGC_v1 replaces the learnable anchors with fixed (non-learnable) anchors, while ANGC_V2 removes the anchor response discrepancy regularization loss $\mathcal{L}_R$. As seen in Table 5, we can find that: 1) compared to ANGC_v1 and ANGC_v2, ANGC achieves ACC performance gains of 10.64% / 3.65%, 6.1% / 1.45%, 8.09% / 5.37%, 18.87% / 2.91%, and 8.79% / 5.35% across the five datasets, respectively, demonstrating that learnable anchors can adaptively capture discriminative substructures and yield more clustering-friendly graph representations. 2) ANGC_v2 suffers noticeable degradation particularly on large datasets, indicating that encouraging diversity among anchor embeddings helps prevent redundancy and enhances representation expressiveness. Overall, the complete ANGC achieves the most robust and superior performance across all datasets, confirming that the combination of learnable anchors and anchor difference regularization is crucial for effective graph-level clustering.

Hyper-Parameter Analysis

Figure 2 illustrates the sensitivity of ANGC to four key hyperparameters on three representative datasets: MUTAG, DD, and COLLAB. The hyperparameters include the number of anchors M, the number of nodes per anchor n, and

Dataset	Metric	ANGC_v1	ANGC_v2	**ANGC**
MUTAG	ACC	74.54±5.18	81.53±4.87	**85.18±3.53**
	NMI	18.59±4.92	36.47±3.15	**39.86±3.58**
	ARI	26.37±2.71	43.46±1.90	**45.61±2.58**
PTC-MR	ACC	52.16±1.22	56.81±1.83	**58.26±1.21**
	NMI	0.32±2.19	**2.73±0.74**	2.65±0.35
	ARI	0.59±0.30	2.13±0.49	**2.73±0.50**
NCI1	ACC	53.39±1.60	56.11±2.24	**61.48±1.91**
	NMI	1.25±0.43	1.41±0.39	**2.67±0.14**
	ARI	1.69±0.23	1.89±0.33	**3.32±0.90**
DD	ACC	56.40±3.82	72.36±3.51	**75.27±4.18**
	NMI	0.12±0.00	18.92±2.11	**20.64±2.21**
	ARI	-0.01±0.00	21.57±1.89	**23.53±2.34**
COLLAB	ACC	55.42±3.70	58.86±3.11	**64.21±2.73**
	NMI	13.02±1.38	17.65±2.87	**26.49±1.56**
	ARI	9.77±1.12	10.74±1.90	**17.31±1.85**

Table 5: Ablation study for the Graph Nyström Anchors. ANGC_v1 employs fixed (non-learnable) anchors in place of learnable ones, and ANGC_v2 removes the anchor response discrepancy regularization loss $\mathcal{L}_R$.

the regularization coefficients β and γ. As shown, ANGC maintains relatively stable performance across a wide range of hyperparameter values, indicating the robustness of our design. For M and n, the accuracy first increases and then slightly decreases, with the optimal range around 20–25, suggesting that a moderate number of anchors and nodes per anchor is sufficient to capture discriminative structural patterns without overfitting. Regarding the regularization parameters β and γ, the performance shows a bell-shaped trend: extremely small or large values lead to performance degradation, while moderate values (e.g., $\beta = 0.01, \gamma = 10$) achieve the best clustering accuracy on MUTAG. These observations clearly confirm that ANGC is not overly sensitive to hyperparameter tuning and can consistently achieve robust and reliable performance across a broad range of parameter settings.

Conclusion

In this work, we propose ANGC, a novel graph-level clustering framework that effectively integrates the expressiveness of kernel methods with the scalability of GNNs via an anchor-driven Nyström approximation. By introducing learnable graph anchors and formulating a response discrepancy regularization, ANGC enables end-to-end optimization of both graph representations and kernel similarities, which enhances the ability of the model to capture fine-grained structural differences across graphs. Extensive experiments on diverse datasets validate the superiority of ANGC in terms of clustering performance and scalability. In future work, we will explore extending the anchor-based framework to hierarchical clustering or multi-view graph scenarios, and further tighten the theoretical understanding of the anchor optimization strategy.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) (Grant No. 62506102, 62562026), the Key Research and Development Program of Hainan Province (Grant No. ZDYF2024GXJS014, ZDYF2023GXJS163), the Collaborative Innovation Project of Hainan University (Grant No. XTCX2022XXB02), and the Natural Science Foundation of Hainan University (Grant No. XJ2400009401).

References

Bai, L.; Cui, L.; Li, M.; Wang, Y.; and Hancock, E. 2024a. QBMK: Quantum-Based Matching Kernels for Un-Attributed Graphs. In *Proceedings of the International Conference on Machine Learning*.

Bai, L.; Xu, Z.; Cui, L.; Li, M.; Wang, Y.; and Hancock, E. 2024b. HC-GAE: The Hierarchical Cluster-based Graph Auto-Encoder for Graph Representation Learning. In *Proceedings of the Conference on Neural Information Processing Systems*, 127968–127986.

Bicciato, A.; Cosmo, L.; Minello, G.; Rossi, L.; and Torsello, A. 2024. GNN-LoFI: a Novel Graph Neural Network through Localized Feature-Based Histogram Intersection. *Pattern Recognition*, 148: 110210.

Borgwardt, K. M.; Gretton, A.; Rasch, M. J.; Kriegel, H.-P.; Schölkopf, B.; and Smola, A. J. 2006. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14): e49–e57.

Borgwardt, K. M.; and Kriegel, H.-P. 2005. Shortest-Path Kernels on Graphs. In *Proceedings of the IEEE International Conference on Data Mining*, 74–81.

Cai, J.; Fan, J.; Guo, W.; Wang, S.; Zhang, Y.; and Zhang, Z. 2022a. Efficient Deep Embedded Subspace Clustering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1–10.

Cai, J.; Wang, S.; Xu, C.; and Guo, W. 2022b. Unsupervised Deep Clustering via Contractive Feature Representation and Focal Loss. *Pattern Recognition*, 123: 108386.

Cai, J.; Zhang, Y.; Fan, J.; Du, Y.; and Guo, W. 2024. Dual Contrastive Graph-Level Clustering with Multiple Cluster Perspectives Alignment. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 3770–3779.

Chen, M.-S.; Lai, P.-Y.; Liao, D.-Z.; Wang, C.-D.; and Lai, J.-H. 2025. Graph Prompt Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cosmo, L.; Minello, G.; Bicciato, A.; Bronstein, M. M.; Rodolà, E.; Rossi, L.; and Torsello, A. 2024. Graph Kernel Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4): 6257–6270.

Deng, B.; Wang, T.; Fu, L.; Huang, S.; Chen, C.; and Zhang, T. 2025. THESAURUS: Contrastive Graph Clustering by Swapping Fused Gromov-Wasserstein Couplings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 16199–16207.

Hu, J.; Xiao, B.; Jin, H.; Duan, J.; Wang, S.; Lv, Z.; Wang, S.; Liu, X.; and Zhu, E. 2025a. SAMCL: Subgraph-Aligned Multiview Contrastive Learning for Graph Anomaly Detection. *IEEE Transactions on Neural Networks and Learning Systems*, 36(1): 1664–1676.

Hu, Y.; Tu, W.; Liu, Y.; Li, M.; Lu, W.; Luo, Z.; Liu, X.; and Chen, P. 2025b. Divide-Then-Rule: a Cluster-Driven Hierarchical Interpolator for Attribute-Missing Graphs. In *Proceedings of the ACM International Conference on Multimedia*, 10905–10914.

Ju, W.; Gu, Y.; Chen, B.; Sun, G.; Qin, Y.; Liu, X.; Luo, X.; and Zhang, M. 2023. Glcc: a General Framework for Graph-Level Clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4391–4399.

Kriege, N. M.; Giscard, P.-L.; and Wilson, R. 2016. On Valid Optimal Assignment Kernels and Applications to Graph Classification. In *Proceedings of the Conference on Neural Information Processing Systems*, 1623–1631.

Lee, J.; Lee, I.; and Kang, J. 2019. Self-Attention Graph Pooling. In *Proceedings of the International Conference on Machine Learning*, 3734–3743.

Li, M.; Micheli, A.; Wang, Y. G.; Pan, S.; Lió, P.; Gnecco, G. S.; and Sanguineti, M. 2024. Guest Editorial: Deep Neural Networks for Graphs: Theory, Models, Algorithms, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4): 4367–4372.

Liang, K.; Meng, L.; Zhou, S.; Tu, W.; Wang, S.; Liu, Y.; Liu, M.; Zhao, L.; Dong, X.; and Liu, X. 2024. Mines: Message Intercommunication for Inductive Relation Reasoning over Neighbor-Enhanced Subgraphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 10645–10653.

Liu, J.; Cheng, J.; Han, R.; Tu, W.; Wang, J.; and Peng, X. 2025a. Federated Graph-Level Clustering Network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 18870–18878.

Liu, J.; Han, R.; Tu, W.; Wang, H.; Wu, J.; and Cheng, J. 2025b. Federated Node-Level Clustering Network with Cross-Subgraph Link Mending. In *Proceedings of the International Conference on Machine Learning*, 38540–38556.

Liu, J.; Liu, X.; Li, C.; Wan, X.; Tan, H.; Zhang, Y.; Liang, W.; Qu, Q.; Feng, Y.; Guan, R.; et al. 2025c. Large-Scale Multi-View Tensor Clustering with Implicit Linear Kernels. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 20727–20736.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. Tudataset: A Collection of Benchmark Datasets for Learning with Graphs. *arXiv preprint arXiv:2007.08663*.

Qian, F.; Cui, L.; Li, M.; Wang, Y.; Du, H.; Xu, L.; Bai, L.; Yu, P. S.; and Hancock, E. R. 2024. AKBR: Learning Adaptive Kernel-Based Representations for Graph Classification. *arXiv preprint arXiv:2403.16130*.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12(9).

Sohn, K. 2016. Improved Deep Metric Learning with Multi-Class N-Pair Loss Objective. In *Proceedings of the Conference on Neural Information Processing Systems*.

Sun, F.-Y.; Hoffman, J.; Verma, V.; and Tang, J. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *Proceedings of the International Conference on Learning Representations*.

Tu, W.; Guan, R.; Zhou, S.; Ma, C.; Peng, X.; Cai, Z.; Liu, Z.; Cheng, J.; and Liu, X. 2024a. Attribute-Missing Graph Clustering Network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 15392–15401.

Tu, W.; Liao, Q.; Zhou, S.; Peng, X.; Ma, C.; Liu, Z.; Liu, X.; Cai, Z.; and He, K. 2024b. RARE: Robust Masked Graph Autoencoder. *IEEE Transactions on Knowledge and Data Engineering*, 36(10): 5340–5353.

Tu, W.; Zhou, S.; Liu, X.; Cai, Z.; Zhao, Y.; Liu, Y.; and He, K. 2025. WAGE: Weight-Sharing Attribute-Missing Graph Autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7): 5760–5777.

Tu, W.; Zhou, S.; Liu, X.; Ge, C.; Cai, Z.; and Liu, Y. 2024c. Hierarchically Contrastive Hard Sample Mining for Graph Self-Supervised Pre-Training. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11): 16748–16761.

Tu, W.; Zhou, S.; Liu, X.; Guo, X.; Cai, Z.; Zhu, E.; and Cheng, J. 2021. Deep Fusion Clustering Network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 9978–9987.

Tu, W.; Zhou, S.; Liu, X.; Liu, Y.; Cai, Z.; Zhu, E.; Zhang, C.; and Cheng, J. 2022. Initializing Then Refining: a Simple Graph Attribute Imputation Network. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 3494–3500.

Vishwanathan, S. V. N.; Schraudolph, N. N.; Kondor, R.; and Borgwardt, K. M. 2010. Graph Kernels. *Journal of Machine Learning Research*, 11: 1201–1242.

Wang, J.; Tang, C.; Zheng, X.; Liu, X.; Zhang, W.; Zhu, E.; and Zhu, X. 2023. Fast Approximated Multiple Kernel k-Means. *IEEE Transactions on Knowledge and Data Engineering*, 36(11): 6171–6180.

Wang, J.; Tu, W.; and Cheng, J. 2025. Hierarchical Shortest-Path Graph Kernel Network. In *Proceedings of the Conference on Neural Information Processing Systems*.

Williams, C.; and Seeger, M. 2000. Using the Nyström Method to Speed Up Kernel Machines. In *Proceedings of the Conference on Neural Information Processing Systems*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *Proceedings of the International Conference on Learning Representations*.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Proceedings of the Conference on Neural Information Processing Systems*.

You, Y.; Chen, T.; Shen, Y.; and Wang, Z. 2021. Graph Contrastive Learning Automated. In *Proceedings of the International Conference on Machine Learning*, 12121–12132.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph Contrastive Learning with Augmentations. In *Proceedings of the Conference on Neural Information Processing Systems*, 5812–5823.

Zaripova, K.; Cosmo, L.; Kazi, A.; Ahmadi, S.-A.; Bronstein, M. M.; and Navab, N. 2023. Graph-in-Graph (GiG): Learning Interpretable Latent Graphs in Non-Euclidean Domain for Biological and Healthcare Applications. *Medical Image Analysis*, 88: 102839.

Zhang, L.; Cosmo, L.; Minello, G.; Torsello, A.; and Rossi, L. 2024. GraFix: A Graph Transformer with Fixed Attention Based on the WL Kernel. In *Proceedings of International Conference on Pattern Recognition*, 435–450.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhang, Y.; Cai, J.; Wu, Z.; Wang, P.; and Ng, S.-K. 2025a. Mixture of Experts as Representation Learner for Deep Multi-View Clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 22704–22713.

Zhang, Y.; Wang, S.; Liu, J.; Yu, S.; Dong, Z.; Liu, S.; Liu, X.; and Zhu, E. 2025b. DLEFT-MKC: Dynamic Late Fusion Multiple Kernel Clustering with Robust Tensor Learning via Min-Max Optimization. In *Proceedings of the International Conference on Learning Representations*.