TowerDNA: Fast and Accurate Graph Retrieval With Dividing, Contrasting and Alignment

Junwei Yang, Yiyang Gu, Yifang Qin, Xiao Luo, Zhiping Xiao, Kangjie Zheng, Wei Ju, *Member, IEEE*, Xian-Sheng Hua, *Fellow, IEEE*, and Ming Zhang, *Member, IEEE*

***Abstract*—Graph retrieval (GR), a ranking procedure that aims to sort the graphs in a database by their relevance to a query graph in decreasing order, has wide applications across diverse domains, such as visual object detection and drug discovery. Existing Graph Retrieval (GR) approaches usually compare graph pairs at a detailed level and generate quadratic similarity scores. In realistic scenarios, conducting quadratic fine-grained comparisons is costly. However, coarse-grained comparisons would result in performance loss. Moreover, label scarcity in real-world data brings extra challenges. To tackle these issues, we investigate a more realistic GR problem, namely, efficient graph retrieval (EGR). Our key intuition is that, since there are numerous underutilized unlabeled pairs in realistic scenarios, by leveraging the additional information they provide, we can achieve speed-up while simplifying the model without sacrificing performance. Following our intuition, we propose an efficient model called Dual-Tower Model with Dividing, Contrasting and Alignment (TowerDNA). TowerDNA utilizes a GNN-based dual-tower model as a backbone to quickly compare graph pairs in a coarse-grained manner. In addition, to effectively utilize unlabeled pairs, TowerDNA first identifies confident pairs from unlabeled pairs to expand labeled datasets. It then learns from remaining unconfident pairs via graph contrastive learning with geometric correspondence. To integrate all semantics with reduced biases, TowerDNA generates prototypes using labeled pairs, which are aligned within both confident and unconfident pairs. Extensive experiments on diverse realistic datasets demonstrate that TowerDNA achieves comparable performance to fine-grained methods while providing a 10× speed-up.**

***Index Terms*—Retrieval, graph neural networks, speed up, dual-tower.**

I. Introduction

GRAPH structure intrinsically lies in various modalities of data, such as images [1], [2], [3], molecules [4], [5], and social networks [6]. Consequently, various graph machine learning problems have received extensive interest in recent years. Within this context, the graph retrieval (GR) problem has been refined and evolved into a significant research area. Graph retrieval aims to rank graphs in a database based on their relevance to a given query graph, which is challenging yet useful.

GR models are typically trained using only the graphs within the database to learn pairwise relevance, and are later used to retrieve the most relevant database graph given an unseen query graph. These approaches have demonstrated remarkable capabilities across various data modalities. For example, GR plays an important role in molecular search within drug discovery [7], [8]. In these applications, molecules are typically represented as graphs, where atoms serve as nodes and chemical bonds as edges. GR algorithms enable the efficient retrieval of novel candidate compounds from chemical databases by identifying those with similar bioactive properties to existing patented drugs, where the patent drugs are treated as query graphs. GR techniques also excel in visual object detection tasks [1], [9], [10]. This process begins with the detection of local feature points in images using methods such as SIFT [11] or CNN-based architectures. These features are then structured into spatial graphs, where nodes correspond to key points and edges encode geometric relationships between these points. The spatial graph extracted from the image to be analyzed is treated as a query graph, which is then matched against a pre-indexed database of object templates to retrieve target objects. Moreover, GR methods have been widely adopted in various other domains, including matching between natural language and programming languages [12], reading comprehension [13], and web server data management [14], and so on.

Previous GR methods typically compute pair-wise interactions with quadratic complexity ($O(|V|^2)$, where $|V|$ denotes the number of nodes) between graph pairs through exhaustive node-level matching (see Fig. 1(a)). In particular, they use deep feature encoders to compute a pair-wise relevance score, which is compared to the ground-truth labels to calculate a loss, being used to optimize the encoders. To achieve better performance, these approaches usually employ sophisticated feature designs to capture fine-grained interactions. For example, they may utilize node-node intersections to facilitate more information exchange [15], [16], employ node-level alignments to obtain fine-grained supervision [1], [17], [18], or leverage path features and multi-scale features to capture higher-order information [19], [20].

Received 3 October 2024; revised 17 August 2025; accepted 30 September 2025. Date of publication 14 October 2025; date of current version 30 December 2025. The work of Ming Zhang was supported by the National Natural Science Foundation of China (NSFC) under Grant 62276002. Recommended for acceptance by Reza Akbarinia. *(Corresponding authors: Xiao Luo; Zhiping Xiao; Ming Zhang.)*

Junwei Yang, Yiyang Gu, Yifang Qin, Kangjie Zheng, Wei Ju, and Ming Zhang are with the School of Computer Science, Peking University, Beijing 100871, China (e-mail: yjwtheonly@pku.edu.cn; yiyanggu@pku.edu.cn; qinyifang@pku.edu.cn; kangjie.zheng@gmail.com; juwei@pku.edu.cn; mzhang_cs@pku.edu.cn).

Xiao Luo is with the Department of Statistics, University of Wisconsin–Madison, Madison, WI 53706 USA (e-mail: xiao.luo@wisc.edu).

Zhiping Xiao is with the Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: patxiao@cs.washington.edu).

Xian-Sheng Hua is with Terminus Group, Beijing 100020, China (e-mail: huaxiansheng@gmail.com).

Digital Object Identifier 10.1109/TKDE.2025.3621493

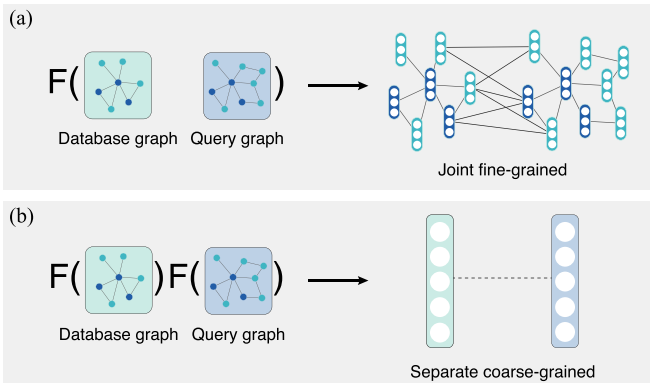

Fig. 1. Fine-grained (a) and coarse-grained (b) GR methods. F is the feature encoder.

Despite their impressive performance, these methods are impractical in most of the realistic applications, due to their slow inference speed. Specifically, because they treat each database-query graph pair as model input, when dealing with a series of queries, they undergo **quadratic model forwarding processes**, where within each forwarding process there involve complex fine-grained (e.g., node-wise) interactions. Moreover, when deploying an optimized GR model in practice, it typically encounters large databases, such as ETKGD [21], which contains 10 M molecular graphs. Therefore, there is an urgent call for faster solutions, so as to make solving realistic applications of GR practical.

To tackle this issue, we study the problem of efficient graph retrieval (EGR). Our goal is to develop a dual-tower (two-tower) model [22], [23], which uses separate encoder (tower) independently to encode graph pairs into a shared representation space and compares graph-level embeddings to generate relevance scores (see Fig. 1(b)). During the inference stage, this approach preprocesses the database into graph-level embeddings and can swiftly respond to multiple queries. This approach involves only **linear model forwarding processes** and quadratic matrix multiplications, which can be efficiently parallelized. However, due to inherent information loss, previous coarse-grained methods almost never achieve comparable performance to fine-grained ones under similar settings [24], [25]. As a matter of fact, we note that in realistic scenarios, annotating all graph pairs or graphs within a large database is costly [13], [26], whereas obtaining unlabeled graphs to expand the database is quite economical. For example, extracting syntactic trees from text [13] or sampling molecular graphs from large-scale SMILES libraries [27], are both easy. The easy accessibility of unlabeled data (graphs or graph pairs) is also a crucial prerequisite for graph-related unsupervised or semi-supervised works [28], [29], yet the unlabeled pairs have not been effectively utilized in previous GR methods.

Hence, in this study, we follow the most widely used semi-supervised setting by assuming that the database (used for GR training) contains a certain amount of unlabeled data, then we aim to address a practical question: **Can we achieve faster inference speed while maintaining superior performance compared to fine-grained models, by integrating coarse-grained models with additional unlabeled data?** If the answer is positive, then when deploying GR models in the future, we can opt for simplified model structures and leverage more unlabeled data to balance speed and performance. To address this question, we propose a new approach called the Dual-**Tower** Model with **D**ividing, Co**n**trasting, and **A**lignment (TowerDNA). Following the dual-tower framework, TowerDNA adopts two separate GNNs to generate embeddings for graph pairs, which are used for rapid calculation of similarity scores. For labeled pairs, we directly compare the similarity scores with the ground truth. For additional unlabeled pairs, we employ a three-stage pipeline to fully leverage their information and compress it into the dual-tower model: **(i) Dividing**: We utilize the momentum memory bank [30] to learn graph prototypes, which are used for dividing unlabeled pairs into confident and unconfident parts. **(ii) Contrasting**: We utilize the learned discriminative embeddings to infer the geometric correspondence of unconfident graph pairs and then use contrastive learning to capture underlying pair-wise information. **(iii) Alignment**: To integrate all semantics with reduced biases, we use prototypes of graphs in labeled pairs as alignment anchors. We encourage graphs in confident pairs to approach their corresponding prototypes by minimizing the entropy of assignments. Additionally, we maintain a memory bank of graphs in unconfident pairs and match their cluster centers with prototypes to provide consistent guidance. We conduct experiments on eight GR datasets constructed from four modalities and find that TowerDNA achieves nearly 10× inference speed-up, while obtaining satisfactory performance. The contributions of this paper are summarized as follows:

- We explore a novel and realistic problem named efficient graph retrieval, which aims to accelerate graph retrieval using additional unlabeled data.
- We introduce a novel model, TowerDNA, which can efficiently compress unlabeled data into a neat dual-tower model and experimental results demonstrate that TowerDNA achieves fast retrieval speed while maintaining satisfactory performance.
- We present an approach that utilizes unlabeled data to trade off model performance and running time. This inference-efficient approach can pave the way for practical implementation of similar models in realistic applications.

II. Related Work

A. Graph Neural Networks

Graph neural networks (GNNs) have demonstrated remarkable capabilities in learning representations from graph-structured data, gaining widespread use across various tasks such as node classification [31], [32] and link prediction [33], [34]. Earlier approaches predominantly relied on spectral GNNs grounded in spectral graph theory [35], [36]. These methods transformed graph signals into embeddings, followed by applying graph Laplacian spectral filters. However, spatial methods have recently overtaken spectral ones due to their reduced computational demands [37], [38]. Spatial GNNs typically follow the message passing framework, where each node aggregates information from its neighbors to iteratively refine its representation. Additionally, GNNs have found extensive application in

graph classification tasks, employing graph pooling functions to convert node-level features into graph-level representations [39], [40]. This has extended the use of GNNs to various practical scenarios, such as molecular property prediction [41], [42], social network analysis [43], and recommendation systems [44], [45]. While these models have shown great promise, their effectiveness often hinges on large amounts of labeled data, which poses challenges in real-world settings where labeled data is scarce and expensive.

B. Graph Retrieval

Graph retrieval (GR) is a significant problem with a variety of practical applications [7], [12], [13]. Early methods [46], [47], [48] mainly rely on manually designed structural similarity metrics to search similar databases graphs for a query graph. With the remarkable development of graph neural networks [38], [49], [50], many neural graph retrieval approaches have been proposed [18], [51], [52] in the last few years. For instance, GMN [53] and SimGNN [16] leverage graph neural networks to obtain meaningful node embeddings and then perform node-level alignment for fine-grained supervision, while ISONET [51] employs an interpretable neural edge alignment mechanism for edge-consistent subgraph matching. However, existing neural GR methods primarily employ the supervised fine-grained learning paradigm, resulting in slow retrieval speeds and neglect of unlabeled data. In contrast, TowerDNA is designed for efficient graph retrieval, featuring a neat model structure and an elaborate pipeline for leveraging unlabeled graph pairs. This enables it to achieve rapid retrieval speeds with satisfactory performance.

C. Semi-Supervised Graph Classification

Since we require unlabeled graph data, some semi-supervised graph methods are relevant to our work. However, there is currently no research on semi-supervised GR, so the closest to our purpose is semi-supervised graph classification, which has gained increasing interest within the machine learning and data mining community [54], [55], [56]. Numerous efforts have been devoted to semi-supervised graph classification for more realistic scenarios in the graph domain, encompassing both node-level classification [57], [58] and graph-level classification [59], [60], [61]. However, these methods cannot be directly applied to GR tasks because they cannot effectively model pair-wise information. Furthermore, they have not explored the potential of utilizing information from additional unlabeled data to achieve speeding-up.

D. Inference Acceleration

Accelerating model inference has become a critical focus in deploying machine learning models, especially in real-world applications where user experience and computational efficiency are paramount. However, speeding up inference often comes with additional costs.

Some approaches require extra training effort. For instance, the Mixture of Experts (MoE) model [62], [63], which is a core technology behind GPT-4 [64], requires meticulously training an additional router to dynamically activate only a subset of model parameters during inference. Another example is non-autoregressive translation (NAT) [65], [66], which achieves parallel inference by predicting all tokens simultaneously instead of sequentially, but this comes at the cost of a significantly more expensive training process to ensure the model performs well despite the lack of token dependencies. Other approaches trade off some performance for faster inference. One such method is model distillation [67], [68], where a larger, more powerful model transfers its knowledge to a smaller, faster model, which may result in lower performance in some domains while retaining high performance in specific tasks. Model pruning [69], [70] is another approach that speeds up inference by removing less important parameters or pruning redundant parts of the model.

Our research is the first to explore this issue in GR problem. We were surprised to find that with only a small amount of additional unlabeled data, we could significantly accelerate inference speed while maintaining, or even exceeding, the performance of conventional models. This approach effectively reduces the typical trade-offs associated with other acceleration methods, providing a practical and efficient solution for real-world applications.

III. TowerDNA for Efficient Graph Retrieval

As our goal is to utilize the information in unlabeled data to balance performance and inference efficiency, this closely aligns with the setting of semi-supervised GR. However, **(i) there is currently no existing research or definition for the semi-supervised GR problem, and (ii) for fast inference, EGR imposes strong constraints on model complexity.** These factors prevent us from utilizing off-the-shelf settings, encouraging us to start our research from scratch. In this section, we formally define the research problem of EGR with unlabeled data and introduce TowerDNA. As introduced in Section I, TowerDNA consists of a labeled module and an unlabeled module. The labeled module employs the dual-tower architecture instantiated with GNNs [71], [72], independently mapping labeled graph pairs to the same latent representation space and calculating similarity based on representation vectors. The unlabeled module comprises three parts: confident-unconfident dividing, contrastive learning, and alignment. The architecture is illustrated in Fig. 2 and we will introduce the details in the following sections.

A. Problem Definition

We denote a graph as $G = (V, E, \mathbf{X})$, where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|V|}] \in \mathbb{R}^{d \times |V|}$ is the node feature matrix, $\boldsymbol{x}_i$ is the node feature vector and d is the feature dimension. In practice, feature vector stores initial node attributes. For example, in a molecular graph, the node vector typically represents the atom type using a one-hot encoding.

In the context of efficient graph retrieval (EGR) with unlabeled data, we have a set of graphs $\mathcal{G} = \mathcal{G}^B \cup \mathcal{G}^Q$. Here, $\mathcal{G}^B = \mathcal{G}^L \cup \mathcal{G}^U$ represents the graph database, consisting of labeled graph set $\mathcal{G}^L = \{G_i^L\}_{i=1}^{|\mathcal{G}^L|}$ and unlabeled graph set $\mathcal{G}^U =$

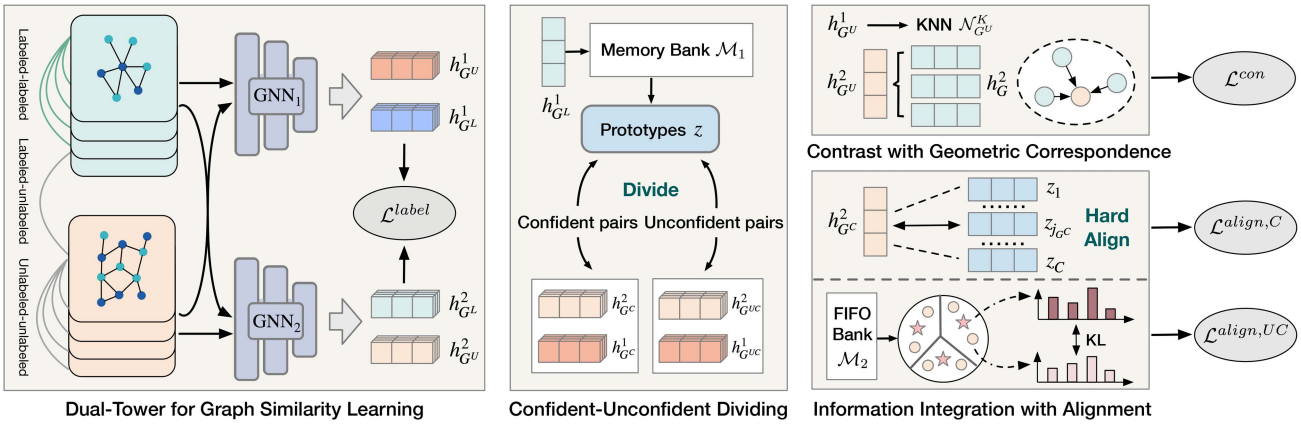

Fig. 2. An overview of the proposed TowerDNA. TowerDNA first employs a GNN-based dual-tower model to generate discriminative graph representations for similarity learning. Then, it divides unlabeled graph pairs into confident and unconfident pairs by memory-based prototype learning. Furthermore, underlying relevance semantics are captured from unlabeled graph pairs by graph contrastive learning with geometric correspondence. Finally, information is integrated from confident and unconfident graph pairs using asymmetric alignment.

$\{G_i^U\}_{i=1}^{|\mathcal{G}^U|}$, where each labeled graph G_i^L belongs to a known semantic category $y_i \in \{1, \ldots, N\}$, where N is the number of categories. The semantic category typically corresponds to an overall property of the graph, such as the toxicity of a drug graph. $\mathcal{G}^Q = \{G_i^Q\}_{i=1}^{|\mathcal{G}^Q|}$ represents the query graph set. The relevance score between two graphs G_1, G_2 is defined as $s(G_1, G_2) \in [-1, 1]$, where a higher score indicates a stronger relevance.

During the training phase, if $G_1 \in \mathcal{G}^L$ and $G_2 \in \mathcal{G}^L$, then $s(G_1, G_2)$ is known and (G_1, G_2) is treated as a labeled pair; otherwise, $s(G_1, G_2)$ is unknown and (G_1, G_2) is a unlabeled pair. During the testing phase, our objective is to rank database $\mathcal{G}^B$ in descending order of relevance scores $s(G^B \in \mathcal{G}^B, G^Q)$ for any query $G^Q \in \mathcal{G}^Q$.

It is important to emphasize that throughout this paper, the terms "labeled" and "unlabeled" refer exclusively to the graphs and graph pairs within the database $\mathcal{G}^B$, since the GR model is trained solely on $\mathcal{G}^B$. Evaluation data, by definition, must have labels and are therefore not the focus of this discussion. Our ultimate goal is to demonstrate that a coarse-grained GR model built on $\mathcal{G}^B \times \mathcal{G}^B$ can achieve faster speed and comparable results compared to a fine-grained GR model built on $\mathcal{G}^L \times \mathcal{G}^L$.

B. Coarse-Grained Graph Similarity Learning

To begin with, we warm up our graph neural network using labeled pairs. In particular, given a labeled graph pair $(G_i^L, G_j^L) \in \mathcal{G}^L \times \mathcal{G}^L$, its ground-truth relevance score is denoted as $s^*(G_i^L, G_j^L)$. To enable TowerDNA to learn the information contained in this supervision signal, we first use a pair of encoders to map them into a shared vector space and calculate their relevance score. We then minimize the difference between this score and $s^*(G_i^L, G_j^L)$.

In particular, our encoders Enc_1 and Enc_2 can adopt any graph neural network approach [71], [72]. We denote the representation of node v in the tth GNN layer as $\boldsymbol{h}_v^{(t)}$, $\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v$. We update $\boldsymbol{h}_v^{(t)}$ iteratively by incorporating information from its associated edges $\{e_{u,v}\}$ and neighbors $\boldsymbol{h}_{\mathcal{N}(v)}^{(t)}$ as follows:

$$\begin{aligned} \boldsymbol{h}_{\mathcal{N}(v)}^{(t)} &= \text{AGG}(\{(\boldsymbol{h}_u^{(t-1)}, e_{u,v}) | u \in \mathcal{N}(v)\}), \\ \boldsymbol{h}_v^{(t)} &= \text{UPDATE}(\boldsymbol{h}_v^{(t-1)}, \boldsymbol{h}_{\mathcal{N}(v)}^{(t)}), \end{aligned} \tag{1}$$

where $\mathcal{N}_v$ denotes the set of neighbors for v. AGG and UPDATE are the aggregation and the update functions of specific GNN architecture. After T iterations, the embedding $\boldsymbol{h}_v^{(T)}$ can embed information of the local subgraph comprising node v's T-hop neighbors. We then apply a READOUT function to get the graph-level representation:

$$\boldsymbol{h}_G = \text{READOUT}(\{\boldsymbol{h}_v^{(T)}\}_{v \in V}). \tag{2}$$

We employ the same GNN architecture in Enc_1 and Enc_2, and denote $\boldsymbol{h}_G^1 = \text{Enc}_1(G)$ and $\boldsymbol{h}_G^2 = \text{Enc}_2(G)$. We then calculate the final relevance score as:

$$s(G_i^L, G_j^L) = \cos(\boldsymbol{h}_{G_i^L}^1, \boldsymbol{h}_{G_j^L}^2), \tag{3}$$

where $\cos(\cdot, \cdot)$ is the cosine similarity. The loss function on labeled pairs $\mathcal{L}^{label}$ is then defined as:

$$\mathcal{L}^{label} = \frac{1}{|\mathcal{G}^L|^2} \sum_{(G_i^L, G_j^L) \in \mathcal{G}^L \times \mathcal{G}^L} (s(G_i^L, G_j^L) - s^*(G_i^L, G_j^L))^2. \tag{4}$$

Here, we employ separate encoders instead of sharing parameters between Enc_1 and Enc_2. Throughout the entire process of TowerDNA, we ensure that Enc_1 is only updated during the computation of $\mathcal{L}^{label}$ on labeled pairs, thereby guaranteeing a more stable similarity calculation, thus, we can utilize Enc_1 to obtain prototypes in Section III-C and constrain Enc_2 in Section III-D. Furthermore, in our experiments (Section IV-E1), we discovered that not sharing parameters leads to better results.

In this labeled module, we avoid complex fine-grained interactions, ensuring faster inference speed. Nonetheless, this approach has its trade-offs, leading to limitations in model

performance. Thus, we leverage additional unlabeled pairs to overcome this issue.

C. Confident-Unconfident Dividing With Memory-Based Prototype Learning

In this section, we aim to divide unlabeled graph pairs into confident and unconfident pairs, thus we can adopt more aggressive methods to utilize reliable information provided by confident pairs and adopt more moderate methods to avoid noise contained in unconfident pairs. However, determining whether a pair can offer confident information is an abstract and challenging task. We adhere to a straightforward intuition: since the relevance scores of labeled pairs are known and confident, if a graph is similar to a labeled graph, the relevance score between this graph and other labeled graphs should be confident as well. To avoid bias that may be introduced by only considering the similarity of a single graph pair, we leverage underlying semantic prototypes to divide confident and unconfident graph pairs.

To be specific, let $\mathcal{G}_i^L$ represent the labeled graph set belonging to the ith category. We first use Enc_1 to calculate the prototype vector $\boldsymbol{z}_i$ for category i:

$$\boldsymbol{z}_i = \frac{1}{|\mathcal{G}_i^L|} \sum_{G_j^L \in \mathcal{G}_i^L} \boldsymbol{h}^1_{G_j^L}. \tag{5}$$

Here, recalculating all $\boldsymbol{h}^1_{G_j^L}$ over $\mathcal{G}_i^L$ in each batch iteration would slow down the training process, while only recalculating a subset of $\boldsymbol{h}^1_{G_j^L}$ would introduce inconsistency among representations due to rapidly changing encoders. Therefore, we employ a memory bank $\mathcal{M}_1$ with momentum updating [30] to address this issue. We initialize $\mathcal{M}_1$ with Enc_1: $\mathcal{M}_1 = \{\boldsymbol{m}_j | \boldsymbol{m}_j = \boldsymbol{h}^1_{G_j^L}, G_j^L \in \mathcal{G}^L\}$, and update $\boldsymbol{m}_j$ after each batch iteration as: $\boldsymbol{m}_j = \alpha \boldsymbol{m}_j + (1-\alpha)\boldsymbol{h}^1_{G_j^L}$, where α is the momentum coefficient and is usually set as 0.3 [30].

The optimized prototype vector for category i using the memory bank is calculated as:

$$\boldsymbol{z}_i = \frac{1}{|\mathcal{G}_i^L|} \sum_{G_j^L \in \mathcal{G}_i^L} \boldsymbol{m}_j. \tag{6}$$

Next, for any unlabeled graph G^U, we examine which prototype is the most related to $\boldsymbol{h}^1_{G^U}$. If the max relevance score exceeds the threshold β, we consider G^U belonging to the confident graph set $\mathcal{G}^C$; otherwise, it belongs to the unconfident graph set $\mathcal{G}^{UC}$. This dividing process can be formalized as:

$$\begin{aligned} \mathcal{G}^C &= \{G^U | \max_i(\cos(\boldsymbol{z}_i, \boldsymbol{h}^1_{G^U})) > \beta, G^U \in \mathcal{G}^U\}, \\ \mathcal{G}^{UC} &= \mathcal{G}^U \; \mathcal{G}^C. \end{aligned} \tag{7}$$

Then, $\mathcal{G}^C \times \mathcal{G}^L$ is considered as the set of confident unlabeled pairs, and $\mathcal{G}^{UC} \times \mathcal{G}^L \cup \mathcal{G}^U \times \mathcal{G}^U$ is considered as the set of unconfident unlabeled pairs. Since $\mathcal{G}^U \times \mathcal{G}^U$ does not contain label information, we utilize graph contrastive learning to model it in Section III-D. Additionally, we employ asymmetric labeled-unlabeled alignment to model $\mathcal{G}^C \times \mathcal{G}^L$ and $\mathcal{G}^{UC} \times \mathcal{G}^L$ in Section III-E to leverage the label information.

D. Graph Contrastive Learning With Geometric Correspondence

For graph pairs in the unconfident set $\mathcal{G}^U \times \mathcal{G}^U$, it is challenging to directly obtain effective supervision from labeled graphs or pairs. Therefore, we conduct contrastive learning within $\mathcal{G}^U \times \mathcal{G}^U$ to enable TowerDNA to capture underlying information in unlabeled pairs.

For each unlabeled graph $G^U \in \mathcal{G}^U$, we first use the graph representation output by Enc_1 to find K graphs with the strongest correspondence, forming the KNN graph set $\mathcal{N}^K_{G^U}$:

$$\begin{aligned} \bar{\mathcal{B}}_{G^U} &= \mathcal{B}_{G^U} - G^U, \\ \mathcal{N}^K_{G^U} &= \{G | \text{top}K(\cos(\boldsymbol{h}^1_G, \boldsymbol{h}^1_{G^U})), G \in \bar{\mathcal{B}}_{G^U}\}, \end{aligned} \tag{8}$$

where $\mathcal{B}_{G^U}$ denotes the batch in which G^U is present during training. Next, we generate positive latent representation $\boldsymbol{h}^{pos}_{G^U}$ with Enc_2 and the KNN graph set $\mathcal{N}^K_{G^U}$:

$$\boldsymbol{h}^{pos}_{G^U} = \frac{1}{K} \sum_{G \in \mathcal{N}^K_{G^U}} \boldsymbol{h}^2_G. \tag{9}$$

We then treat all graphs in $\mathcal{B}_{G^U}$ except G^U itself as negative samples and calculate the contrastive learning loss [73] $\mathcal{L}^{con}$ as:

$$\begin{aligned} r_{G^U}(\boldsymbol{h}) &= \exp(\boldsymbol{h} \cdot \boldsymbol{h}^2_{G^U}/\tau), \\ \bar{\mathcal{B}}_{G^U} &= \mathcal{B}_{G^U} - G^U, \\ \mathcal{L}^{con} &= \frac{1}{|\mathcal{G}^U|} \sum_{G^U} -\log \frac{r_{G^U}(\boldsymbol{h}^{pos}_{G^U})}{r_{G^U}(\boldsymbol{h}^{pos}_{G^U}) + \sum_{G \in \bar{\mathcal{B}}_{G^U}} r_{G^U}(\boldsymbol{h}^2_G)}, \end{aligned} \tag{10}$$

where $r_{G^U}(\boldsymbol{h})$ is used for calculating similarity between $\boldsymbol{h}$ and G^U's representation encoded by Enc_2, τ is the temperature parameter and is set as 0.1 following [73].

During contrastive learning, we utilize the KNN found by Enc_1 to guide the optimization of Enc_2. If they share parameters, this approach resembles self-training [74], where the model is guided by its own outputs, potentially leading to error accumulation. On the other hand, when they do not share parameters, this approach is akin to co-training [75]. Here, Enc_1 influences Enc_2 through KNN, while Enc_2 affects Enc_1 through $\mathcal{L}^{label}$ (4), thereby reducing error accumulation. In this process, by encouraging similar graphs to converge in the representation space, our method implicitly performs clustering, resulting in a more organized and compact representation space.

E. Information Integration With Alignment

In this section, we aim to align unlabeled graphs to labeled graphs, to facilitate modeling of the relevance information contained in $\mathcal{G}^U \times \mathcal{G}^L$ pairs. Here, we adopt two asymmetric methods for aligning confident and unconfident pairs, respectively.

Confident Alignment: First, we deal with confident pair set $\mathcal{G}^C \times \mathcal{G}^L$. For graphs in the confident unlabeled graph set $\mathcal{G}^C$,

we explore two different alignment strategies. One is a hard alignment strategy similar to pseudo-labeling. For any $G^C \in \mathcal{G}^C$, we find the prototype $\boldsymbol{z}_{j_{G^C}}$ (calculated in (6)), which is the most relevant to G^C's representation $\boldsymbol{h}^2_{G^C}$. We then set the relevance score between G^C and all graphs from category j_{G^C} to 1. And set the score between G^C and other categories to -1. We calculate the alignment loss $\mathcal{L}_1^{align,C}$ similar to (4):

$$j_{G^C} = \arg\max_i(\cos(\boldsymbol{z}_i, \boldsymbol{h}^2_{G^C})),$$

$$\mathcal{L}_1^{align,C} = \frac{1}{|\mathcal{G}^C|}\sum_{G^C}(\cos(\boldsymbol{z}_{j_{G^C}}, \boldsymbol{h}^2_{G^C}) - 1)^2 + \frac{1}{|\mathcal{G}^C|}\sum_{G^C}\sum_{i,i\neq j_{G^C}}(\cos(\boldsymbol{z}_i, \boldsymbol{h}^2_{G^C}) + 1)^2. \quad (11)$$

Here, inspired by PRISM [76], we compute the relevance loss between G^C and prototypes rather than individual graphs to avoid quadratic complexity. However, this hard alignment may introduce noise when G^C has similar relevance to prototypes from multiple classes. Therefore, we design a soft alignment strategy. We normalize the relevance score and treat them as probabilities:

$$p_{G^C,j} = \frac{\cos(\boldsymbol{z}_j, \boldsymbol{h}^2_{G^C}) + 1 + \epsilon}{\sum_i \cos(\boldsymbol{z}_i, \boldsymbol{h}^2_{G^C}) + 1 + \epsilon}, \quad (12)$$

here, $p_{G^C,j}$ can be regarded as the probability that the graph G^C belongs to the category j, and ϵ is a small value to prevent singularities. We then encourage $\boldsymbol{h}^2_{G^U}$ to get close to fewer categories by minimizing the entropy:

$$\mathcal{L}_2^{align,C} = \frac{1}{|\mathcal{G}^C|}\sum_{G^C} -\sum_j p_{G^C,j}\log(p_{G^C,j}). \quad (13)$$

Unconfident Alignment: Then, we deal with unconfident pair set $\mathcal{G}^{UC} \times \mathcal{G}^L$. For graphs in the unconfident unlabeled set $\mathcal{G}^{UC}$, aligning each graph directly to a specific semantic category, as done to the confident set $\mathcal{G}^C$, might introduce errors due to the data distribution difference. Benefiting from the KNN based contrastive learning in the previous section, $\mathcal{G}^{UC}$ has already undergone implicit clustering. Thus, we can leverage cluster information to alleviate noise and bias caused by aligning a single graph.

To be specific, we attempted explicit KMEANS clustering [77] on the representation space $\{\boldsymbol{h}^2_{G^{UC}} | G^{UC} \in \mathcal{G}^{UC}\}$. During this process, we find that clustering once at the beginning of each epoch always provides outdated results, while clustering once per batch iteration significantly slowed down training. Therefore, we utilized a first-in first-out memory bank $\mathcal{M}_2$ to address this issue. $\mathcal{M}_2$ maintains a set of representation vectors with a size not exceeding M. Before each gradient descent iteration, the representation vectors of the current batch are placed into $\mathcal{M}_2$. If the storage limit is reached, the oldest vectors are removed. Subsequently, KMEANS is performed only on the vectors in $\mathcal{M}_2$ instead of $\{\boldsymbol{h}^2_{G^{UC}} | G^{UC} \in \mathcal{G}^{UC}\}$. Once the clustering results are obtained, let $f(G^{UC}) \in \mathbb{N}$ denote the cluster id corresponding to graph G^{UC}, $count_j$ is the number of vectors contained in cluster j, we first calculate the center vector $\boldsymbol{c}_j$ for cluster j, and transform the relevance score to probabilities:

$$\boldsymbol{c}_j = \frac{1}{count_j}\sum_{G\in\mathcal{M}_2, f(G)=j}\mathcal{M}_2(G),$$

$$q_{j,i} = \frac{\cos(\boldsymbol{z}_i, \boldsymbol{c}_j) + 1 + \epsilon}{\sum_k \cos(\boldsymbol{z}_k, \boldsymbol{c}_j) + 1 + \epsilon},$$

$$p_{G^{UC},i} = \frac{\cos(\boldsymbol{z}_i, \boldsymbol{h}^2_{G^{UC}}) + 1 + \epsilon}{\sum_k \cos(\boldsymbol{z}_k, \boldsymbol{h}^2_{G^{UC}}) + 1 + \epsilon}, \quad (14)$$

where $q_{j,i}$ can be approximated as the probability that cluster j belongs to category i, and $p_{G^{UC},i}$ can be approximated as the probability that graph G^{UC} belongs to category i. Then we guide the alignment of unconfident graphs using the relationship between cluster center $\boldsymbol{c}_j$ and all labeled prototype vectors $\{\boldsymbol{z}_k\}$ as:

$$\mathcal{L}^{align,UC} = \frac{1}{|\mathcal{G}^{UC}|}\sum_{G^{UC}} KL([q_{f(G^{UC}),1}, \ldots, q_{f(G^{UC}),N}] || [p_{G^{UC},1}, \ldots, p_{G^{UC},N}]). \quad (15)$$

Here, N is the number of categories, $KL(\cdot||\cdot)$ represents the KL divergence, leading to the consistency of underlying category distribution between single graph G^{UC} and cluster $f(G^{UC})$. It is worth noting that this is essentially a form of soft alignment, allowing multiple clusters to correspond to the same underlying category. Therefore, the number of clusters set in KMEANS can be greater than the actual number of categories in $\mathcal{M}_2$. In our experiments, we directly set the number of clusters to N.

F. Training and Inference

During the training phase, we first warm up our network using the labeled pairs as in (4). Then, we optimize our model by minimizing the following objective:

$$\mathcal{L} = \mathcal{L}^{label} + \gamma_1\mathcal{L}^{con} + \gamma_2\mathcal{L}^{align,C} + \gamma_3\mathcal{L}^{align,UC}. \quad (16)$$

In each iteration, in addition to sampling graph pair set $\mathcal{B}^{pair}$, we also sample unlabeled graphs from $\mathcal{G}^U$ to form set $\mathcal{B}^U$. We calculate the loss function within these two sets and update the model parameters accordingly. Afterward, we update the memory bank $\mathcal{M}_1$ using the representations of graphs involved in $\mathcal{B}^{pair}$ and update $\mathcal{M}_2$ using the representations of graphs in $\mathcal{B}^U$. Next, we analyze the time and memory complexity during each training iteration. For simplicity, we assume all graphs have approximately the same number of nodes and edges, denoted as $|V|$ and $|E|$, respectively.

- The time and memory complexity for encoding all graph representations are both $O((|\mathcal{B}^U| + |\mathcal{B}^L|) \times (|V| + |E|))$.
- The time and memory complexity for computing $\mathcal{L}^{label}$ are $O(|\mathcal{B}^L|)$.
- Updating $\mathcal{M}_1$ and generating prototype vectors causes a time complexity of $O(|\mathcal{B}^L|)$ and a memory complexity of $O(|\mathcal{G}^L|)$.
- The time and memory complexity for dividing confident and unconfident pairs are $O(|\mathcal{B}^U| \times N)$, where N is the number of categories.

- The time and memory complexity for computing the contrastive loss $\mathcal{L}^{con}$ are $O(|\mathcal{B}^U|^2)$.
- The time and memory complexity for computing $\mathcal{L}^{align,C}$ are $O(|\mathcal{B}^U| \times N)$.
- The time complexity for computing $\mathcal{L}^{align,UC}$ is $O(iter \times |\mathcal{M}_2| \times N + |\mathcal{B}^U| \times N)$, and its memory complexity is $O(|\mathcal{M}_2| \times N + |\mathcal{B}^U| \times N)$, where $iter$ denotes the number of K -MEANS iterations (set to 10in our experiments).

Overall, the time complexity for a single batch iteration is:

$$O((|\mathcal{B}^U| + |\mathcal{B}^L|) \times (|V| + |E|) + |\mathcal{B}^U| \times N + |\mathcal{B}^U|^2 + iter \times |\mathcal{M}_2| \times N).$$

The space complexity is:

$$O((|\mathcal{B}^U| + |\mathcal{B}^L|) \times (|V| + |E|) + |\mathcal{B}^U| \times N + |\mathcal{B}^U|^2 + |\mathcal{M}_2| \times N + |\mathcal{G}^L|).$$

During the inference phase, we use Enc_1 to encode the database graphs and use Enc_2 to encode the query graphs. **We then directly sort the database based on the cosine similarities of the obtained vectors**. Thus, the overall time and memory complexity of calculating similarities in inference phase are both $O((|\mathcal{G}^B| + |\mathcal{G}^Q|) \times (|V| + |E|) + |\mathcal{G}^B| \times |\mathcal{G}^Q|))$. We list a streamlined training and inference process in Algorithm 1.

Algorithm 1: Learning and Inference Algorithm of TowerDNA.

Input: Graph database $\mathcal{G}^B = \mathcal{G}^L \cup \mathcal{G}^U$, labeled pair relevance score $\{s(G_1, G_2) | (G_1, G_2) \in \mathcal{G}^L \times \mathcal{G}^L\}$, a query graph G^Q
Parameter: Dual-tower encoder Enc_1 and Enc_2
Output: Ranked database with decreasing relevance to G^Q.

Train Enc_1 and Enc_2 on labeled pairs with (4).
  $Iter_{max} \leftarrow$ Convergence.
Retrain Enc_1 and Enc_2 for $0.2Iter_{max}$ iterations with (4).
Initialize $\mathcal{M}_1$ with $\text{Enc}_1(\mathcal{G}^L)$
Initialize $\mathcal{M}_2$ with $\text{Enc}_2(\mathcal{G}^U)$
while not convergence **do**
  Sample labeled graph pairs $\mathcal{B}^{pair}$
  $\mathcal{B}^L \leftarrow$ graphs involved in $\mathcal{B}^{pair}$
  Sample unlabeled graphs $\mathcal{B}^U$
  $\mathcal{B} \leftarrow \mathcal{B}^L \cup \mathcal{B}^U$
  $\mathcal{L}^{label} \leftarrow \mathcal{B}^{pair}, s(\cdot, \cdot)$
  Dividing // Section III-C
    Prototype $\boldsymbol{z} \leftarrow \mathcal{M}_1$
    $\mathcal{B} \times \mathcal{B} = \mathcal{B}^L \times \mathcal{B}^L \cup \mathcal{B}^L \times \mathcal{G}^C \cup \mathcal{B}^L \times \mathcal{G}^{UC} \cup \mathcal{B}^U \times \mathcal{B}^U \leftarrow$ Prototype $\boldsymbol{z}$
  Contrastive learning // Section III-D
    $\mathcal{L}^{con} \leftarrow \mathcal{B}^U \times \mathcal{B}^U$
  Alignment // Section III-E
    $\mathcal{L}^{align,C} \leftarrow \mathcal{B}^L \overset{align}{\longleftrightarrow} \mathcal{G}^C$
    Cluster center $\boldsymbol{c} \leftarrow \mathcal{M}_2$
    Distribution $P_1 \leftarrow \boldsymbol{c} \overset{align}{\longleftrightarrow} \boldsymbol{z}$
    Distribution $P_2 \leftarrow \mathcal{G}^{UC} \overset{align}{\longleftrightarrow} \mathcal{B}^L$
    $\mathcal{L}^{align,UC} \leftarrow KL(P_1, P_2)$
  $\mathcal{L} = \mathcal{L}^{label} + \gamma_1 \mathcal{L}^{con} + \gamma_2 \mathcal{L}^{align,C} + \gamma_3 \mathcal{L}^{align,UC}$ // (16)
  Update Enc_1 and Enc_2 with $\mathcal{L}$
  $\mathcal{M}_1 \leftarrow \text{Enc}_1(\mathcal{B}^L)$
  $\mathcal{M}_2 \leftarrow \text{Enc}_2(\mathcal{B}^U)$
end while
Ranked database $\leftarrow \cos(\text{Enc}_1(G^B), \text{Enc}_2(G^Q)))$

IV. Experiments

A. Dataset

We conduct our experiments on eight graph datasets extracted from four completely different modalities of data sources (1) **Molecules**, i.e., AIDS and MUTAG; (2) **Bioinformatics**, i.e., PROTEINS and ENZYMES; (3) **Computer vision**, i.e., COIL-DEL and MSRC9; (4) **Social networks**, i.e., IMDB-BINARY and IMDB-MULTI. These datasets are publicly accessible [78] and the statistics are listed in Table I. More details of these datasets are introduced as follows:

- *AIDS [4]:* Each graph in the AIDS dataset corresponds to a molecular compound from the AIDS Antiviral Screen Database [79], which is labeled according to its bioactivity against HIV. We download this dataset from AIDS.
- *MUTAG [5]:* Each graph in the MUTAG dataset represents a chemical compound, which is annotated according to its mutagenic effect on a bacterium. We download this dataset from MUTAG.
- *PROTEINS [80]:* Each graph in the PROTEINS dataset corresponds to the secondary structure of a protein, which is labeled as a category among helix, sheet, and turn. We download this dataset from PROTEINS.
- *ENZYMES [80]:* Each graph in the ENZYMES dataset corresponds to the tertiary structure of a protein, which is labeled as a category of top-level Enzyme Commission. We download this dataset from ENZYMES.
- *COIL-DEL [4]:* Each graph in the COIL-DEL dataset is created using corner features extracted from an image, which is labeled with a category indicating the object present in the image. We download this dataset from COIL-DEL.
- *MSRC9 [81]:* Each graph in the MSRC9 dataset is constructed by extracting superpixels from an image, which is annotated with a semantic category of the specific object depicted in the image. We download this dataset from MSRC9.
- *IMDB-BINARY & IMDB-MULTI [6]:* Each graph in the IMDB-BINARY and IMDB-MULTI datasets is an ego-network reflecting the movie-collaboration relationships of an actor or actress, which is labeled with a specific class of movie genres. We download datasets from IMDB-BINARY and IMDB-MULTI.

Since the graphs in these datasets exhibit distinctly different topological and feature characteristics covering multiple realistic senarios (Fig. 3), evaluating performance on these datasets, we can demonstrate that the proposed graph retrieval setup is

TABLE I
STATISTICS OF EIGHT GRAPH DATASETS

Statistics	MUTAG	AIDS	ENZYMES	PROTEINS	COIL-DEL	MSRC9	IMDB-B	IMDB-M
Num. of graphs	188	2000	600	1113	3900	221	1000	1500
Avg. nodes	17.93	15.69	32.63	39.06	21.54	40.58	19.77	13.00
Avg. edges	19.79	16.20	62.14	72.82	54.24	97.94	13.00	65.94
Sampled unlabeled graphs	56	600	180	333	1170	67	300	450
Sampled labeled graphs	56	600	180	333	1170	67	300	450
Labeled pairs	3136	360000	32400	110889	1368900	4489	90000	202500
Unlabeled pairs	9408	1080000	97200	332667	4106700	13467	270000	607500
Test pairs	4256	480000	43200	149184	1825200	5762	120000	270000

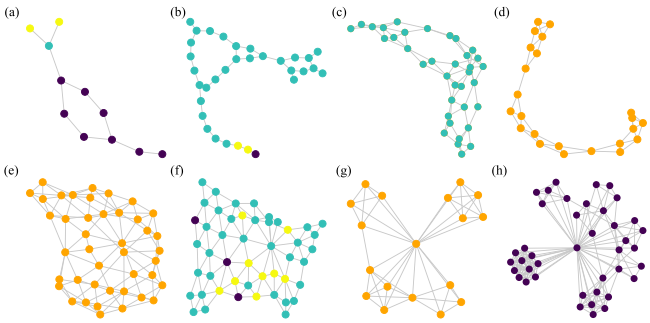

Fig. 3. Graph samples from MUTAG (a), AIDS (b), ENZYMES (c), PROTEINS (d), COIL-DEL (e), MSRC9 (f), IMDB-B (g) and IMDB-M (h).

naturally aligned with several real-world applications where the goal is to determine whether two graphs share meaningful semantic properties. Concretely, TowerDNA has the potential to serve as a powerful tool across a variety of domains:

- *Drug Discovery:* In molecular biology, molecules are naturally represented as graphs of atoms and bonds. Graph retrieval enables efficient querying of molecular databases to identify compounds that share bioactivity—a semantic label—even when they are not identical at the atomic level. For example, given a molecular graph with promising activity against a target (e.g., HIV), semantic retrieval can uncover candidate molecules that exhibit analogous bioactivity properties, even without knowing the exact structural motifs responsible for the activity. This capability facilitates lead compound identification, scaffold hopping, and drug repurposing [82].
- *Object Detection:* In computer vision, graphs can be constructed from object features in images (e.g., superpixels in MSRC9). Retrieval in this setting allows matching objects across images based on semantic similarity of their general graph representations, even under changes in local structure, pose, or background [83]. This corresponds exactly to predicting whether two object graphs encode the same semantic class.
- *Social Network Analysis:* In social networks, users and their interactions can be modeled as graphs. Retrieval in this setting makes it possible to identify ego-networks with similar structural patterns, which corresponds to predicting whether two user groups share the same semantic label in terms of social role. For instance, in movie collaboration networks like IMDB-BINARY and IMDB-MULTI, retrieval allows identification of user groups (e.g., actors, producers) with similar interaction dynamics, enabling genre-specific marketing strategies or automated cluster labeling.

All graphs in these datasets are labeled with semantic categories. We adapt them to the EGR setting following SSH [84], the semi-supervised image retrieval task. We randomly divide each dataset into 3 parts: the graph database, validation query graph set, and test query graph set with proportions of 60%, 20%, and 20%, respectively. We train TowerDNA on the graph database. Within the database, we randomly select 50% of the graphs as unlabeled graphs, and their labels are treated as unknown throughout training to rigorously evaluate our model's capability in handling unlabeled data. The relevance scores between labeled graphs are known in the training phase, while the rest graph pairs remain unlabeled. We follow a commonly used setting in retrieval-related research to derive pairwise relevance scores from semantic labels [84], [85], [86], [87]. For a labeled graph pair (or a database-query pair), the relevance score is set to 1 if two graphs share at least one semantic label provided by original dataset; otherwise, the score is set to -1.

B. Compared Methods

As mentioned in Section III, there is no off-the-shelf method available as a baseline for EGR, we adapt two categories of methods to address this problem:

Coarse-grained methods: Since a subset of graphs in the database can be assigned with semantic labels, we aim to train a classification model on adapted EGR datasets and then use the optimized model to solve the EGR problem. As these methods only independently encode graph pairs and perform similarity calculations at the graph level, we categorize them as coarse-grained methods.

To leverage unlabeled data, we select four semi-supervised classification models:

- *GCN∗:* It uses pseudo-labeling to implement a semi-supervised variants of GCN [72], which is the most classic model for graph-structured learning, which generates the normalized adjacency matrix to generate discriminative node representations.
- *GAT∗:* It uses pseudo-labeling to implement a semi-supervised variants of GAT [71], which utilizes the attention mechanism to generate adaptive weights for neighborhood aggregation.

- *InfoGraph [59]:* It compares node representations and graph representations by mutual information maximization to generate discriminative graph representations.
- *ASGN [88]:* It is a semi-supervised graph classification method, which utilizes a teacher-student architecture to distill abundant semantic knowledge from unlabeled graphs and merges it with labeled information.

We explore two approaches to adapt these models for EGR:

- *Classification* adaptation: For a query graph, the *Classification* first assigns it to a specific class c using a classifier. Subsequently, *Classification* calculates the likelihood of all database graphs belonging to the class c. These graphs are then ranked in descending order based on the likelihood.
- *Feature* adaptation: *Feature* method utilizes the features before the classification head as representation vectors and calculates the relevance scores using cosine similarity or L2 distance.

Finally, we select the best adaptation method and relevance score calculation method from above two approaches based on validation performance.

Fine-grained methods: We adopt graph similarity based methods for graph retrieval. We chose following six methods as baselines:

- *SimGNN [16]:* It transforms graph samples into embeddings, which would be enhanced using the attention mechanism. Then it generates graph similarity scores by comparing embeddings.
- *GMN [53]:* It proposes a graph matching network that takes a pair of graphs as input and computes a similarity score between them. GMN computes the similarity score jointly on the pair rather than independently mapping each graph to a vector.
- *ERIC [52]:* It obtains graph learning with a node-graph correspondence constraint for training high-quality GNNs, followed by a multi-scale graph edit distance estimation to enhance the expressive ability.
- *ISONET [51]:* It uses a new scoring function to get asymmetric relevance metrics and learns the edge-consistent mapping function to identify the matched subgraph.
- *SNA [89]:* It introduces a self-supervised nodewise attention-guided framework that employs correlation-guided contrastive learning to capture discriminative node embeddings.
- *ISONET++ [90]:* It proposes an early-interaction GNN with cross-graph message passing guided by injective node alignment. It also conducts node-pair partner interaction that exploits edge existence/non-existence signals for performance refinement.

As mentioned previously, these graph similarity works are fine-grained methods. Similarly, we employ the pseudo-labeling technique to adapt them for modeling unlabeled pairs. The adapted models were denoted as SimGNN*, GMN*, ERIC*, ISONET*, SNA* and ISONET++*.

We compare the above-mentioned models with our TowerDNA, examining their retrieval Mean Average Precision (MAP) and retrieval speed. Additionally, we also compare a simple variant of our model, TowerDNA^{hard}, which employs hard alignment (11) instead of soft alignment (13).

C. Implementation Details

All methods are implemented in PyTorch. We adopt GCN to instantiate encoders in TowerDNA since GCN * outperforms GAT* in our experiments (Table II). Following SimGNN, the embedding dimension is set to 64, the learning rate is set to 5e-3, and the weight decay is set to 0. Following K-CLUSTER [91], in K MEANS algorithm, the distance metric is cosine similarity and the max iteration is set to 10. The cluster centers are initialized with prototypes obtained from $\mathcal{M}_1$, adding a Gaussian noise. We set the threshold β to 0.8 and randomly sample 32 labeled graph pairs and 32 unlabeled graphs to form $\mathcal{B}^{pair}$ and $\mathcal{B}^{U}$, respectively. In the KNN algorithm in contrastive learning, we set K to 3. The size of the memory bank M is set to $\max(192, 5N)$, where N represents the number of semantic categories. The hyper-parameters γ_1 is chosen from (0.3,0.5,0.7), and set $\gamma_2 = \gamma_3 = (1-\gamma_1)/2$. The selection criteria of γ_1, γ_2, and γ_3 are based on the model's performance on the validation query set. Finally, we use Adam to optimize all methods. The source code is available at: https://github.com/yjwtheonly/TowerDNA.

For fairness and reproducibility, we implement GAT and GCN using PyTorch Geometric's standardized modules. All other baselines are evaluated using their official codebases: InfoGraph, ASGN, SimGNN, GMN, ERIC, ISONET, SNA, and ISONET++. All models are trained and evaluated on the same server to eliminate performance differences caused by hardware variations. Specifically, all experiments use a single NVIDIA GeForce RTX 3090 GPU (24 GB), running on Ubuntu 18.04.6 LTS with an Intel Xeon CPU E5-2697 v4 @ 2.30 GHz and 512 GB of RAM. During inference, we set the batch size to 128 for all models. To avoid the impact of data loading, we preload all data into memory before measuring inference time. We repeat inference twice and record the average time. We sum the total inference time of each model across all datasets and designate the fastest model as the base model (which is GCN$*$ in our experiments). We then calculate the relative inference time of all other models with respect to this base model. For example, the relative inference time of TowerDNA is $2\times$ in Table II, meaning its total inference time is approximately twice that of the base model.

D. Results and Analysis

We compare all the aforementioned models on eight datasets. We use MAP to measure model performance and use inference time to measure retrieval speed. The results are presented in Table II. Through our analysis, we draw the following three conclusions:

1) TowerDNA Can Efficiently Address the EGR Problem: First, because the quadratic interaction occurs solely in the final embedding space, our proposed TowerDNA achieves inference speeds comparable to other coarse-grained methods, rendering it **10 times faster** than other fine-grained methods. Furthermore, as shown in Table III, the training time of TowerDNA is on par with other fine-grained models.

Moreover, while maintaining a high retrieval speed, TowerDNA achieves **comparable performance** to methods with fine-grained pairwise interactions (SimGNN*, GMN*,

TABLE II
THE OVERALL MAP AND INFERENCE TIME OF DIFFERENT METHODS. FINE INDICATES WHETHER THE METHOD UTILIZES FINE-GRAINED INTERACTIONS. UNLABELED INDICATES WHETHER IT INCORPORATES UNLABELED DATA, AND * INDICATES THE METHOD IS ADJUSTED TO MODEL UNLABELED DATA USING PSEUDO-LABELING. RELATIVE TIME IS THE TOTAL INFERENCE TIME DIVIDED BY THAT OF THE FASTEST MODEL. THE BEST RESULTS ARE **BOLD**, AND THE BEST RESULTS IN THE COARSE-GRAINED MANNER ARE UNDERLINED.

Methods	Fine	Unlabeled	Inference time (s)	Relative time	MUTAG	AIDS	ENZYMES	PROTEINS	COIL-DEL	MSRC9	IMDB-B	IMDB-M
GCN*		✓	131	1×	0.623	0.932	0.195	0.640	0.183	0.793	0.520	0.339
GAT*		✓	153	1×	0.610	0.941	0.201	0.623	0.174	0.782	0.512	0.310
InfoGraph		✓	320	2×	0.615	0.897	0.183	0.612	0.185	0.819	0.513	0.351
ASGN		✓	504	4×	0.642	0.913	0.213	0.630	0.192	0.784	0.510	0.343
SimGNN	✓		3050	23×	0.633	0.960	0.202	0.621	0.171	0.832	0.501	0.332
GMN	✓		3800	29×	0.602	0.932	0.204	0.603	0.165	0.794	0.493	0.341
ERIC	✓		2117	16×	0.691	0.922	0.226	0.653	0.201	0.819	0.506	0.350
ISONET	✓		2865	22×	0.641	0.951	0.214	0.631	0.212	0.842	0.493	0.345
SNA	✓		5429	41×	0.671	0.957	0.204	0.635	0.218	0.844	0.497	0.324
ISONET++	✓		4457	34×	0.682	0.964	0.224	0.657	0.221	0.859	0.497	0.314
SimGNN*	✓	✓	3023	23×	0.693	0.982	0.229	0.654	0.192	0.869	0.517	0.346
GMN*	✓	✓	3789	29×	0.631	0.947	0.220	0.632	0.172	0.821	0.523	0.353
ERIC*	✓	✓	2126	16×	0.710	0.961	0.251	0.704	0.221	0.873	**0.563**	**0.382**
ISONET *	✓	✓	2880	22×	0.672	0.974	0.233	0.664	0.224	0.863	0.514	0.339
SNA *	✓	✓	5416	41×	0.684	0.971	0.213	0.657	0.231	0.857	0.509	0.347
ISONET++ *	✓	✓	4463	34×	0.693	0.979	**0.254**	0.697	0.235	**0.884**	0.516	0.342
TowerDNA^{hard}		✓	276	2×	0.703	**0.985**	0.225	**0.710**	0.231	0.853	0.521	0.356
TowerDNA		✓	255	2×	**0.712**	0.976	0.243	0.680	**0.253**	0.877	0.519	0.374

TABLE III
RELATIVE TRAINING TIME OF DIFFERENT MODELS

Model	SimGNN*	GMN*	ERIC*	ISONET*	SNA*	ISONET++*	TowerDNA
Relative training time	4×	2×	1×	2×	5×	4×	2×

ERIC*, ISONET*, SNA* and ISONET++*). TowerDNA and TowerDNA^{hard} exhibit the best results on MUTAG, AIDS, PROTEINS and COIL-DEL, with no significant difference from other fine-grained methods on the remaining three datasets. Additionally, we observe thatTowerDNA^{hard} outperforms TowerDNA on three datasets with only two categories. This could be attributed to the fact that the hard alignment approach does not introduce significant noise when there are only a few categories present.

We further investigate whether TowerDNA is capable of implicitly capturing fine-grained structural similarities between graph pairs, even when trained solely with graph-level supervision. We leverage Graph Edit Distance (GED)-based similarity to evaluate this. GED measures the minimum number of operations—such as node or edge insertions, deletions, and substitutions—required to transform one graph into another. Following the definition in SimGNN, we compute the GED-based relevance score between two graphs G_1 and G_2 as: $\text{Relevance}(G_1, G_2) = \exp(-\frac{2 \cdot \text{GED}(G_1, G_2)}{|G_1|+|G_2|}) \in (0, 1]$. This continuous score reflects the fine-grained structural similarity between graph pair. For each query graph, we first use the trained GR model (e.g., TowerDNA) to rank all graphs in the database based on semantic-guided relevance, producing a ranking sequence s_1. Next, we compute GED-based continuous relevance scores between the same query and database graphs, and sort the graphs accordingly to obtain a ranking sequence s_2. We treat s_2 as the "golden standard" and assess how well s_1 aligns with it by computing MAP. As shown in Table IV, TowerDNA achieves a high degree of consistency with GED-based rankings. Moreover, its final performance is comparable to that of fine-grained methods, indicating that TowerDNA can effectively model underlying structural similarities even with only graph-level supervision.

TABLE IV
ALIGNMENT BETWEEN SEMANTIC-GUIDED RETRIEVAL AND GED-BASED RETRIEVAL. THE GED-BASED RANKING SEQUENCE IS TREATED AS THE GOLDEN STANDARD, AND THE MAP SCORE IS CALCULATED FOR EACH MODEL'S RETRIEVAL RESULTS ACCORDINGLY.

Methods	MUTAG	AIDS	MSRC9
GCN*	0.523	0.692	0.625
ERIC*	0.614	0.793	0.752
SNA*	0.653	0.741	0.746
ISONET++*	0.612	0.782	0.731
TowerDNA	0.632	0.764	0.723

In summary, our experimental results demonstrate that TowerDNA and its variant not only effectively but also efficiently solve EGR.

2) Leveraging Unlabeled Data for Performance and Speed Enhancement is Feasible: By evaluating the performance of SimGNN, GMN, ERIC, ISONET and their respective variants, we observe significant performance improvements when employing pseudo-labeling techniques to incorporate additional unlabeled data. Comparing the four variants and TowerDNA, we notice that the former adopts an intricate model with a simple unlabeled data modeling framework, while the latter employs a simple model with an elaborate unlabeled data modeling framework. Both frameworks demonstrate similar performance in solving GR problems, but TowerDNA exhibits notably faster inference speeds. Considering realistic scenarios with abundant unlabeled data and a need for high-speed inference, the latter framework proves more suitable. In other words, by effectively utilizing the information in unlabeled data, we can enhance the inference speed of GR models while maintaining comparable

TABLE V
RESULTS OF TOWERDNA AND SEVERAL VARIANTS ENCODING DATABASE AND QUERY WITH DIFFERENT ENCODERS

Methods	Database	Query	MUTAG	PROTEINS	MSRC9
TowerDNA^{21}	Enc_2	Enc_1	0.653	0.630	0.823
TowerDNA^{11}	Enc_1	Enc_1	0.641	0.627	0.821
TowerDNA^{22}	Enc_2	Enc_2	0.709	0.685	0.862
TowerDNA^{single}	Enc	Enc	0.682	0.640	0.844
TowerDNA	Enc_1	Enc_2	0.712	0.680	0.877

performance, which is crucial for practical applications of GR models.

3) Adapting Graph Classification to Graph Retrieval is Not a Trivial Problem: From the empirical aspect, a model performing well in classification might not necessarily perform well in EGR. As shown in Table II, despite the strong performance of InfoGraph and ASGN in classification tasks in various studies [59], [92], [93], their superiority does not stand out when dealing with EGR problems.

From the theoretical aspects, there are several reasons behind this phenomenon. First, classification results depend solely on the class with the highest probability, making continuous changes in probability distribution could cause a sudden shift in categories. Consequently, adaptations relying on classification results face substantial penalties in case of misclassifications. We can intuitively grasp these penalties through a simplified version of *Classification* adaptation, denoted as *Classification*^{s}. *Classification*^{s} classifies both database graphs and queries into a specific category. Database graphs with the same category as the query are ranked higher, while the other graphs are ranked lower. Assuming the accuracy of the adopted classifier is $p \in [0, 1]$, two intriguing findings emerge: (i) When there are numerous potential categories, *Classification*^{s} can correctly identify a pair of relationships with the probability of merely p^2. (ii) In the case of only two categories, using classifiers with accuracy $1 - p$ and p respectively can achieve the same MAP. These phenomena indicate the substantial risks associated with classification-based retrieval. Second, the classification does not involve explicit pairwise comparisons during training. This fundamental difference poses challenges when adapting these methods to the retrieval problem. As a result, *Feature* adaptation struggles to identify an effective similarity metric in the feature space.

E. Ablation Study and Analysis

In this section, we explore several variants of TowerDNA to address the following four questions:

1) What Roles Do the Two Encoders Play Individually?: Since we employ an unconventional approach, specifically using two separate encoders instead of one shared encoder, we will provide a deeper analysis of this approach in this section from both empirical and analytical perspectives.

Empirically, we conduct an ablation study to investigate the roles of each encoder in the dual-tower structure by modifying the training and inference methods. As shown in Table V, TowerDNA ^{single} employs only one encoder Enc during both training and inference, while the other four methods maintain the training phase unchanged and use different encoders to encode the query and database during the reference phase. We can observe that TowerDNA^{11} performs the poorest because Enc_1 solely explicitly models labeled data and can only indirectly access unlabeled information through calculating $\mathcal{L}^{label}$ from Enc_2. TowerDNA^{21} exhibits inferior performance because Enc_2 encodes unlabeled data during training, making it more effective at encoding unseen query graphs. The effectiveness of TowerDNA^{22} and TowerDNA is comparable, possibly because Enc_2 learns sufficient pairwise information through contrastive learning (Section III-D). TowerDNA^{single} performs worse than TowerDNA, as analyzed in Section III-D, the reason is during contrastive learning, Enc utilizes representations encoded by itself to find K NNs, which then guide its own contrastive learning, potentially leading to error accumulation.

Analytically, we explore the superiority of using two encoders in TowerDNA by analyzing its relationship with co-training and mean-teacher methods.

(i) Relation to co-training: Co-training [75] is a well-known semi-supervised learning method that uses separate models, trains them independently, and assigns labels to each other's outputs to avoid error accumulation caused by using one model. In our case, if we were to use only one encoder Enc in TowerDNA, and denote the KNN set found by Enc as $\mathcal{N}^K$, this set itself would be considered as positive samples. Training on these samples following (10) would lead to the accumulation and reinforcement of errors produced by Enc itself. In contrast, by using two encoders, Enc_1 and Enc_2, to find the KNN sets for a given graph G, denoted as $\mathcal{N}_G^{K,1}$ and $\mathcal{N}_G^{K,2}$ respectively, and training Enc_2 with $\mathcal{N}_G^{K,1}$ as positive samples, we effectively apply a variation of co-training. In this process, the similarity calculations between G and $G' \in \mathcal{N}^{K,1} \cap \mathcal{N}^{K,2}$ by Enc_2 will converge more rapidly, while the similarity calculations for G and $G' \in \mathcal{N}^{K,1} \cup \mathcal{N}^{K,2} - \mathcal{N}^{K,1} \cap \mathcal{N}^{K,2}$ will fluctuate. This implies that error accumulation occurs only when Enc_1 and Enc_2 incorrectly select the positive samples simultaneously. *(ii) Relation to mean-teacher:* Mean-teacher [94] is another well-known semi-supervised learning method, where a weaker but more stable teacher model guides the training of a student model. In our case, Enc_1 can be seen as the teacher model, trained only on labeled data, making it more stable. By constraining Enc_2 with the KNN results from Enc_1, we help reduce the instability when training on unlabeled data.

2) How Does the Proportion of Unlabeled Graphs Affect Performance?: While leveraging unlabeled data is central to our framework, understanding the relationship between data efficiency and model performance is crucial for practical applications. This ablation study investigates how different proportions of unlabeled graphs impact TowerDNA's effectiveness. We systematically vary the usage proportion of unlabeled graphs from 0% to 100% while keeping the labeled data fixed, comparing TowerDNA's performance across different settings with six fine-grained baselines (SimGNN, GMN, ERIC, ISONET, SNA, and ISONET++) that donot use unlabeled data. As shown in Fig. 4, without unlabeled data, TowerDNA underperforms all fine-grained methods – an expected outcome since TowerDNA is fundamentally a coarse-grained method lacking fine structural

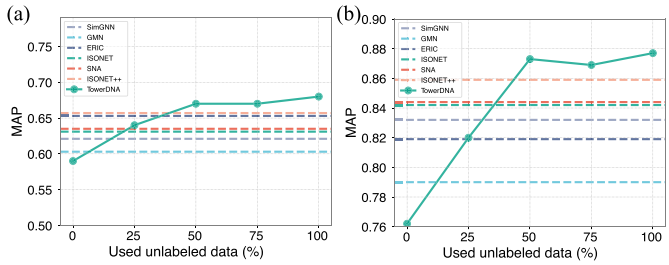

Fig. 4. Results of TowerDNA when using different rates of unlabeled graphs on PROTEINS (a) and MSRC9 (b).

TABLE VI
ABLATION STUDY OF VARIANTS REMOVING DIFFERENT MODULES FOR UNLABELED DATA

Methods	MUTAG	PROTEINS	MSRC9	Relative training time
TowerDNA^{-div}	0.692	0.643	0.868	1.4×
TowerDNA^{-con}	0.624	0.613	0.812	1.1×
TowerDNA^{-align}	0.707	0.663	0.860	1.0×
TowerDNA	0.712	0.680	0.877	1.3×

matching capability. However, with merely 25% additional unlabeled graphs, TowerDNA achieves performance comparable to fine-grained baselines, while maintaining a inference speed approximately 10× faster. This transition indicates that TowerDNA framework can effectively convert limited unlabeled data into discriminative structural knowledge, compensating for the inherent information loss in coarse-grained matching.

3) Are All Three Modules for Unlabeled Pairs Effective?: This ablation study investigates the influence of removing each unlabeled module. We evaluate the performance of three variants of TowerDNA. TowerDNA^{-div} does not conduct the dividing process and treats all unlabeled pairs as unconfident pairs. TowerDNA^{-con} skips the contrastive learning. And TowerDNA^{-align} skips the alignment process. As shown in Table VI, method TowerDNA^{-con} performs the worst because the contrastive learning actually aligns Enc_1 and Enc_2, and removing this module may lead to discrepancies between them. Furthermore, contrastive learning implicitly clusters graphs by minimizing distances to their nearest neighbors, ensuring the effectiveness of subsequent alignments for unconfident pairs. TowerDNA^{-div} exhibits subpar performance, indicating that treating all pairs as unconfident pairs may not effectively utilize the information contained in labels. TowerDNA^{-align} is worse than TowerDNA, indicating that our proposed alignment approach can efficiently leverage the confident information and underlying semantic guidance. We further analyze the efficiency of each variant. Since all variants behave identically during inference, Table VI only reports relative training time. Among them, TowerDNA^{-align} achieves the fastest training speed, indicating that the alignment module is the most time-consuming component—primarily due to the K-MEANS clustering involved. Nevertheless, the differences in training time among all variants are relatively small. This suggests that the dominant computational cost lies in the graph encoding step performed by the encoders, rather than in the specific loss modules.

TABLE VII
ABLATION STUDY ABOUT MEMORY BANKS

Methods	AIDS	ENZYMES	COIL-DEL
TowerDNA$^{-\mathcal{M}_1}$	0.953	0.199	0.212
TowerDNA$^{-\mathcal{M}_2}$	0.951	0.213	0.231
TowerDNA	0.976	0.243	0.253

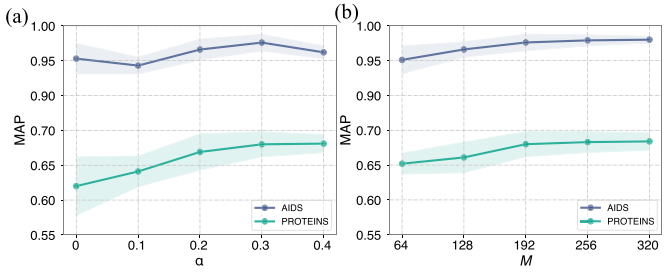

Fig. 5. Sensitivity analysis about momentum coefficient (a) and memory bank size (b).

4) Is the Use of Two Memory Banks Necessary?: This ablation study explores the effect of the momentum memory bank $\mathcal{M}_1$ and first-in-first-out memory bank $\mathcal{M}_2$. As shown in table VII, TowerDNA$^{-\mathcal{M}_1}$ removes $\mathcal{M}_1$, which means calculating prototypes using representation generated in each batch directly. TowerDNA$^{-\mathcal{M}_2}$ removes $\mathcal{M}_2$, which means cluster and calculate the cluster centers at the beginning of each epoch. We can observe significant performance drops in both variants. The TowerDNA$^{-\mathcal{M}_1}$ suffers from discontinuous representations caused by rapidly changing encoders. The TowerDNA$^{-\mathcal{M}_2}$ is affected because the cluster information cannot be updated promptly.

F. Sensitivity Analysis

We further investigate the influence of momentum coefficient α and memory bank size M on the performance of our TowerDNA, as shown in Fig. 5. We first vary momentum coefficient α in the range of $\{0, 0.1, 0.2, 0.3, 0.4\}$. As we can see, the performance of the model improves gradually till saturation as the momentum coefficient rises since smoother updates can facilitate consistency of graph representations, benefiting robust prototype learning. Then, we vary memory bank size M in the range of $\{64, 128, 192, 256, 320\}$ with all other hyper-parameters fixed. It can be observed that a larger memory bank results in better retrieval performance before saturation, which is beneficial from sufficient and representative graph representations provided for clustering and alignment.

For the threshold β and the number of neighbors K, we determined the hyperparameters for all datasets solely based on the validation performance on the PROTEINS dataset. The results are shown in Table VIII. From Table VIII, we can observe that when β is too small, the model tends to generate more confident pairs, which introduces excessive noisy data and leads to a substantial drop in performance. Ultimately, we set $\beta = 0.8$ for all datasets. We can further observe that when K is too small, the performance is poor. However, when $K \geq 3$, there

TABLE VIII
PERFORMANCE OF USING DIFFERENT THRESHOLD β AND DIFFERENT K IN KNN ON PROTEINS

β	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
MAP	0.61	0.61	0.62	0.63	0.63	0.66	0.68	0.71	0.68
Confident(%)	100	99.1	96.3	88.2	83.4	62.5	43.8	24.3	15.6

K	1	2	3	4	5
MAP	0.63	0.65	0.71	0.69	0.69

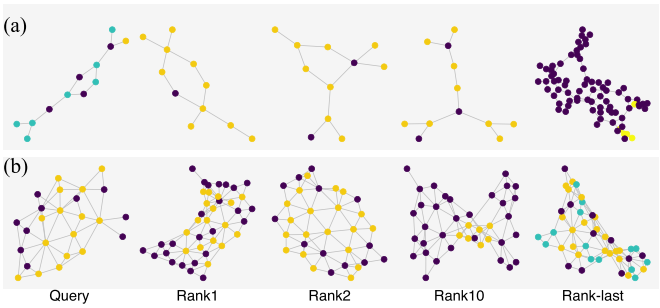

Fig. 6. Illustration of retrieval results on two datasets AIDS (a) and MSRC9 (b).

is no significant difference in performance. Therefore, we chose $K = 3$ for all datasets.

G. Case Study

We demonstrate the power of TowerDNA by visualizing the retrieval results. In Fig. 6, we present two cases of TowerDNA on AIDS (a) and MSRC9 (b). On AIDS, the query graph and the top two ranked results exhibit similar cyclic structures, contrasting with significant structural differences in the graph with the worst rank. On MSRC9, the top 1st, 2nd, and 10th ranked results share identical node features with the query graph, in contrast to the distinctive node features present in the graph ranked at the bottom. This illustration indicates that TowerDNA not only models semantic similarities but also effectively captures structure and feature similarities.

V. CONCLUSION

This paper studies a realistic and underexplored graph retrieval problem, termed efficient graph retrieval (EGR), and proposes a novel model named Dual-**Tower** Model with **D**ividing, Co**n**trasting and **A**lignment (TowerDNA). The core idea of TowerDNA is compressing additional unlabeled data into a neat model, thus achieving both fast retrieval speed and satisfactory performance. TowerDNA employs a GNN-based dual-tower model to generate coarse-grained representations for fast retrieval. TowerDNA also uses a dividing, contrasting, and alignment pipeline to fully leverage abundant unlabeled graph pairs. Extensive empirical results demonstrate that TowerDNA can achieve both high retrieval speed and satisfactory performance on realistic EGR datasets.

ACKNOWLEDGMENT

The authors are grateful to the anonymous reviewers for their careful reading and for giving important suggestions to improve this article.

REFERENCES

[1] T. Yu, R. Wang, J. Yan, and B. Li, "Deep latent graph matching," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12187–12197.
[2] C. Schellewald and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation," in *Proc. Int. Workshop Energy Minimization Methods Comput. Vis. Pattern Recognit.* Springer, 2005, pp. 171–186.
[3] T. Yu et al., "Generalizing graph matching beyond quadratic assignment model," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 861–871.
[4] K. Riesen and H. Bunke, "Iam graph database repository for graph based pattern recognition and machine learning," in *Proc. Struct., Syntactic, Stat. Pattern Recognit.: Joint IAPR Int. Workshop*, Orlando, USA, 2008, pp. 287–297.
[5] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *J. Med. Chem.*, vol. 34, no. 2, pp. 786–797, 1991.
[6] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1365–1374.
[7] W. Zhao, D. Zhou, B. Cao, K. Zhang, and J. Chen, "Adversarial modality alignment network for cross-modal molecule retrieval," *IEEE Trans. Artif. Intell.*, vol. 5, no. 1, pp. 278–289, Jan. 2024.
[8] A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé, and G. Pujadas, "Molecular fingerprint similarity search in virtual screening," *Methods*, vol. 71, pp. 58–63, 2015.
[9] J. Johnson et al., "Image retrieval using scene graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3668–3678.
[10] Y. Gu, Y. Wang, and Y. Li, "A survey on deep learning-driven remote sensing image scene understanding: Scene classification, scene retrieval and scene-guided object detection," *Appl. Sci.*, vol. 9, no. 10, 2019, Art. no. 2110.
[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.
[12] X. Ling et al., "Deep graph matching and searching for semantic code retrieval," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 5, pp. 1–21, 2021.
[13] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000 questions for machine comprehension of text," 2016, *arXiv:1606.05250*.
[14] M. Aldwairi, M. Jarrah, N. Mahasneh, and B. Al-khateeb, "Graph-based data management system for efficient information storage, retrieval and processing," *Inf. Process. Manage.*, vol. 60, no. 2, 2023, Art. no. 103165.
[15] L. Wang, Y. Zheng, D. Jin, F. Li, Y. Qiao, and S. Pan, "Contrastive graph similarity networks," *ACM Trans. Web*, vol. 18, pp. 1–20, 2023.
[16] Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, 2019, pp. 384–392.
[17] A. Zanfir and C. Sminchisescu, "Deep learning of graph matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2684–2693.
[18] K. D. Doan, S. Manchanda, S. Mahapatra, and C. K. Reddy, "Interpretable graph similarity computation via differentiable optimal alignment of node embeddings," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 665–674.
[19] X. Ling et al., "Multilevel graph matching networks for deep graph similarity learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 799–813, Feb. 2023.
[20] K. Goyal, U. Gupta, A. De, and S. Chakrabarti, "Deep neural matching models for graph retrieval," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1701–1704.
[21] S. Riniker and G. A. Landrum, "Better informed distance geometry: Using what we know to improve conformation generation," *J. Chem. Inf. Model.*, vol. 55, no. 12, pp. 2562–2574, 2015.
[22] Y. Yu, W. Wang, Z. Feng, and D. Xue, "A dual augmented two-tower model for online large-scale recommendation," *DLP-KDD*, 2021.
[23] J. Yang et al., "Mixed negative sampling for learning two-tower neural networks in recommendations," in *Proc. Companion Proc. Web Conf.*, 2020, pp. 441–447.

[24] L. Su et al., "Beyond two-tower matching: Learning sparse retrievable cross-interactions for recommendation," in *Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2023, pp. 548–557.

[25] S. Wang, J. Chang, Z. Wang, H. Li, W. Ouyang, and Q. Tian, "Fine-grained retrieval prompt tuning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 2644–2652.

[26] C. G. Specht et al., "Quantitative nanoscopy of inhibitory synapses: Counting gephyrin molecules and receptor binding sites," *Neuron*, vol. 79, no. 2, pp. 308–321, 2013.

[27] G. Zhou et al., "Uni-mol: A universal 3D molecular representation learning framework," in *Proc. 11th Int. Conf. Learn. Representations*, Kigali, Rwanda, May 2023. [Online]. Available: https://openreview.net/forum?id=6K2RM6wVqKu

[28] X. Yang, Z. Song, I. King, and Z. Xu, "A survey on deep semi-supervised learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 8934–8954, Sep. 2023.

[29] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020.

[30] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9729–9738.

[31] G. Guo et al., "Learning adaptive node embeddings across graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 6028–6042, Jun. 2022.

[32] T. Zhao, X. Zhang, and S. Wang, "GraphSMOTE: Imbalanced node classification on graphs with graph neural networks," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 833–841.

[33] L. Cai, J. Li, J. Wang, and S. Ji, "Line graph neural networks for link prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5103–5113, Sep. 2022.

[34] Z. Zhu, Z. Zhang, L.-P. Xhonneux, and J. Tang, "Neural bellman-ford networks: A general graph neural network framework for link prediction," in *Adv. Neural Inf. Process. Syst. 34: Annu. Conf. Neural Inf. Process. Syst.*, Dec. 2021, pp. 29476–29490. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/f6a673f09493afcd8b129a0bcf1cd5bc-Abstract.html

[35] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[36] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*.

[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Representations*, Toulon, France, Apr. 2017.

[38] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks," in *Proc. 7th Int. Conf. Learn. Representations*, New Orleans, LA, USA, May 2019.

[39] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 4805–4815.

[40] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3734–3743.

[41] H. Stärk et al., "3D infomax improves GNNs for molecular property prediction," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20479–20 502.

[42] J. Yang et al., "Mol-AE: Auto-encoder based molecular representation learning with 3D cloze test objective," 2024, *bioRxiv 2024-04*.

[43] X. Li, L. Sun, M. Ling, and Y. Peng, "A survey of graph neural network based recommendation in social networks," *Neurocomputing*, vol. 549, 2023, Art. no. 126441.

[44] I. Kumar, Y. Hu, and Y. Zhang, "Eflec: Efficient feature-leakage correction in GNN based recommendation systems," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 1885–1889.

[45] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: A survey," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–37, 2022.

[46] H. Jiang, H. Wang, S. Y. Philip, and S. Zhou, "Gstring: A novel approach for efficient search in graph databases," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, 2006, pp. 566–575.

[47] H. He and A. K. Singh, "Closure-tree: An index structure for graph queries," in *Proc. 22nd Int. Conf. Data Eng.*, 2006, pp. 38–38.

[48] Y. Liang and P. Zhao, "Similarity search in graph databases: A multi-layered indexing approach," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 783–794.

[49] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst. 30: Annu. Conf. Neural Inf. Process. Syst.*, pp. 1024–1034, Long Beach, CA, USA, Dec. 2017.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[51] I. Roy, V. S. B. R. Velugoti, S. Chakrabarti, and A. De, "Interpretable neural subgraph matching for graph retrieval," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 8115–8123.

[52] W. Zhuo and G. Tan, "Efficient graph similarity computation with alignment regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 30181–30193.

[53] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli, "Graph matching networks for learning the similarity of graph structured objects," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3835–3845.

[54] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 529–536.

[55] D.-H. Lee et al., "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. Workshop Challenges Representation Learn.*, 2013, Art. no. 896.

[56] B. Zhang et al., "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 18408–18419.

[57] W. Feng et al., "Graph random neural networks for semi-supervised learning on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 22092–22103.

[58] P. Li, Y. Yang, M. Pagnucco, and Y. Song, "CoGNet: Cooperative graph neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2022, pp. 1–8.

[59] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," 2019, *arXiv: 1908.01000*.

[60] Y. Xie, Y. Liang, M. Gong, A. Qin, Y.-S. Ong, and T. He, "Semisupervised graph neural networks for graph classification," *IEEE Trans. Cybern.*, vol. 53, no. 10, pp. 6222–6235, Oct. 2023.

[61] W. Ju et al., "Tgnn: A joint semi-supervised framework for graph-level classification," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, 2022, pp. 2122–2128.

[62] Z. Chen, Y. Deng, Y. Wu, Q. Gu, and Y. Li, "Towards understanding mixture of experts in deep learning," 2022, *arXiv:2208.02813*.

[63] T. Chen et al., "Adamv-moe: Adaptive multi-task vision mixture-of-experts," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 17346–17 357.

[64] J. Achiam et al., "GPT-4 technical report," 2023, *arXiv:2303.08774*.

[65] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher, "Non-autoregressive neural machine translation," 2017, *arXiv: 1711.02281*.

[66] Y. Xiao et al., "A survey on non-autoregressive generation for neural machine translation and beyond," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 11407–11427, Oct. 2023.

[67] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, 2021.

[68] J. Wang, W. Bao, L. Sun, X. Zhu, B. Cao, and S. Y. Philip, "Private model compression via knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1190–1197.

[69] X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 21702–21720.

[70] J. Liu et al., "Discrimination-aware network pruning for deep model compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4035–4051, Aug. 2022.

[71] P. Velickovic et al., "Graph attention networks," *Stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

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

[73] F. Wang and H. Liu, "Understanding the behaviour of contrastive loss," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2495–2504.

[74] Y. Zou, Z. Yu, X. Liu, B. Kumar, and J. Wang, "Confidence regularized self-training," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5982–5991.

[75] W. Wang and Z.-H. Zhou, "A new analysis of co-training," in *Proc. Int. Conf. Mach. Learn.*, 2010, Art. no. 3.

[76] C. Liu et al., "Noise-resistant deep metric learning with ranking-based instance selection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6811–6820.

[77] J. MacQueen, "Classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.

[78] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," 2020, *arXiv: 2007.08663*.

[79] D. DTP, "Aids antiviral screen," 2004. [Online]. Available: http://dtp.nci.nih.gov/docs/aids/aids-data.html
[80] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.
[81] M. Neumann, N. Patricia, R. Garnett, and K. Kersting, "Efficient graph kernels by randomization," in *Proc. Mach. Learn. Knowl. Discov. Databases: Eur. Conf.*, Bristol, U.K., 2012, pp. 378–393.
[82] S. Doshi and S. P. Chepuri, "A computational approach to drug repurposing using graph neural networks," *Comput. Biol. Med.*, vol. 150, 2022, Art. no. 105992.
[83] M. Manzo, S. Pellino, A. Petrosino, and A. Rozza, "A novel graph embedding framework for object recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 341–352.
[84] S. Zhang, J. Li, and B. Zhang, "Pairwise teacher-student network for semi-supervised hashing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 730–737.
[85] Y. Feng, H. Zhu, D. Peng, X. Peng, and P. Hu, "Rono: Robust discriminative learning with noisy labels for 2D-3D cross-modal retrieval," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 11610–11619.
[86] D. Jin et al., "Cgmn: A contrastive graph matching network for self-supervised graph similarity learning," 2022, *arXiv:2205.15083*.
[87] L. Wang, Y. Zheng, D. Jin, F. Li, Y. Qiao, and S. Pan, "Contrastive graph similarity networks," *ACM Trans. Web*, vol. 18, no. 2, pp. 1–20, 2024.
[88] Z. Hao et al., "ASGN: An active semi-supervised graph neural network for molecular property prediction," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 731–752.
[89] G. Wen et al., "Exploring attention and self-supervised learning mechanism for graph similarity learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 6, pp. 11106–11120, Jun. 2024.
[90] A. Ramachandran, V. Raj, I. Roy, S. Chakrabarti, and A. De, "Iteratively refined early interaction alignment for subgraph matching based graph retrieval," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 77593–77629.
[91] K. P. Sinaga and M.-S. Yang, "Unsupervised K-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80716–80727, 2020.
[92] W. Ju et al., "TGNN: A joint semi-supervised framework for graph-level classification," 2023, *arXiv:2304.11688*.
[93] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, vol. 8, pp. 193907–193934, 2020.
[94] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1195–1204.

Junwei Yang received the BS degree in computer science from Peking University, Beijing, China, in 2022. He is currently working toward the PhD degree in computer science. His research interests include graph machine learning, large language model, AI for science and bioinformatics.

Yiyang Gu received the BS degree in computer science from Peking University, Beijing, China, in 2021. He is currently working toward the PhD degree in computer science. His research interests include data mining, graph machine learning, self-supervised learning, and bioinformatics.

Yifang Qin recieved the BS degree in computer science from Peking University, Beijing, China, in 2023. He is currently working toward the PhD degree in computer science. His research interests include graph representation learning and recommender systems.

Xiao Luo is an assistant professor in the Department of Statistics, University of Wisconsin–Madison, Madison, USA. His research interests include machine learning, AI for science, data mining and bioinformatics. He has published more than 120 papers in refereed journals and conference proceedings such as NeurIPS, ICML, ICLR, CVPR, ICCV, ACL, NAACL, EMNLP, TheWebConf, ICDE, SIGKDD, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Image Processing*, and *Bioinformatics*. He is also an editorial board member of BMC Bioinformatics.

Zhiping Xiao received the BS degree in Peking University, Beijing, China, in 2016, the MS degree in computer science from University of California at Berkely, in 2018, and the PhD degree in computer science from the University of California at Los Angeles, in 2024. She is currently a postdoc researcher in the Paul G. Allen School of Computer Science & Engineering, University of Washington at Seattle. Her research interests include graph representation learning, graph neural networks, and applications such as social network analysis and computational biomedicine. She won the Best Paper Award of the Annual Meeting of the Association for Computational Linguistics (ACL) 2025 as a co-author.

Kangjie Zheng received the BS degree in computer science from the Harbin Institute of Technology, Harbin, China, in 2020. He is currently working toward the PhD degree in school of computer science, Peking University. His research interests include text generation, AI for science and bioinformatics.

Wei Ju (Member, IEEE) received the BS degree in mathematics from Sichuan University, Sichuan, China, in 2017 and the PhD degree in computer science from Peking University, Beijing, China, in 2022. He is currently a postdoc research fellow in Computer Science with Peking University. His current research interests lie primarily in the area of machine learning on graphs including graph representation learning and graph neural networks, and interdisciplinary applications such as knowledge graphs, bioinformatics, drug discovery and recommender systems. He has published more than 40 papers in top-tier venues and has won the best paper finalist in IEEE ICDM 2022.

Xian-Sheng Hua (Fellow, IEEE) received the BS and PhD degrees in applied mathematics from Peking University, Beijing, China, in 1996 and 2001, respectively. In 2001, he joined Microsoft Research Asia, Beijing, as a researcher. He has been a senior researcher with Microsoft Research Redmond, Redmond, WA, USA, since 2013. He became a Researcher and the Senior Director with the Alibaba Group, Hangzhou, China, in 2015. He has authored or coauthored more than 250 research papers and has filed more than 90 patents. His research interests include the areas of multimedia search, advertising, understanding, and mining, pattern recognition, and machine learning. Dr. Hua is an ACM Distinguished Scientist. He has served on the Technical Directions Board of the IEEE Signal Processing Society. He was a recipient of the MIT Technology Review Innovators Under 35 Asia Pacific (MIT35). He served as the Program co-chair for the ACM Multimedia 2012, the IEEE International Conference on Multimedia and Expo (ICME) 2012, and the IEEE ICME 2013.

Ming Zhang (Member, IEEE) received the BS, MS, and PhD degrees in computer science from Peking University, Beijing, China, in 1988, 1991, and 2005, respectively. She is currently a full professor with the School of Computer Science, Peking University. She has authored or coauthored more than 200 research papers on text mining and machine learning in the top journals and conferences. She is a leading author of several textbooks on data structures and algorithms in Chinese and the corresponding course is awarded as the National Elaborate Course, National Boutique Resource Sharing Course, National Fine-designed Online Course, and National First Class Undergraduate Course by Ministry of Education (MOE) China. Prof. Zhang is a member of the Advisory Committee of the Ministry of Education in China. She is one of the 15 members of the ACM/IEEE CC2020 Steering Committee. She received the Best Paper Award of the International Conference on Machine Learning (ICML) 2014, the Best Paper Nominee of World Wide Web (WWW) 2016 and the Best Paper Award of the Annual Meeting of the Association for Computational Linguistics (ACL) 2025 as the corresponding author.