Motif-Driven Contrastive Learning of Graph Representations

Shichang Zhang, Ziniu Hu, *Member, IEEE*, Arjun Subramonian, and Yizhou Sun

***Abstract*—Pre-training Graph Neural Networks (GNN) via self-supervised contrastive learning has recently drawn lots of attention. However, most existing works focus on node-level contrastive learning, which cannot capture global graph structure. The key challenge to conduct subgraph-level contrastive learning is to sample informative subgraphs that are semantically meaningful. To solve it, we propose to learn graph motifs, which are frequently-occurring subgraph patterns (e.g. functional groups of molecules), for better subgraph sampling. Our framework MotIf-driven Contrastive leaRning Of Graph representations (*MICRO-Graph*) can: 1) use GNNs to extract motifs from large graph datasets; 2) leverage learned motifs to sample informative subgraphs for contrastive learning of GNN. We formulate motif learning as a differentiable clustering problem, and adopt EM-clustering to group similar and significant subgraphs into several motifs. Guided by these learned motifs, a sampler is trained to generate more informative subgraphs, and these subgraphs are used to train GNNs through graph-to-subgraph contrastive learning. By pre-training on the ogbg-molhiv dataset with *MICRO-Graph*, the pre-trained GNN achieves 2.04% ROC-AUC average performance enhancement on various downstream benchmark datasets, which is significantly higher than other state-of-the-art self-supervised learning baselines.**

***Index Terms*—Data mining, graph neural network.**

I. Introduction

GRAPH-STRUCTURED data, such as molecules and social networks, are ubiquitous in many research areas and real-world applications. Recently, Graph Neural Networks (GNNs) have shown great expressive power for learning graph representations without explicit feature engineering [1], [2], [3], [4]. To empower GNNs to capture graph structural and semantic properties without human annotations, a line of works has been proposed to pre-train GNNs with self-supervised learning (SSL) [5], [6], [7], [8], [9], [10], [11], [12], [13]. The pre-trained GNNs could generalize to downstream tasks within the same domain (e.g., all molecules in the chemical domain) and enhance performance with fine-tuning.

Many GNN pre-training works could be categorized into the contrastive learning framework, which forces views from the same data instance (e.g., different crops from an image or different nodes from a graph) to become closer and pushes views from different instances apart. One key component in contrastive learning is to generate informative and diverse views from each data instance. For example, in computer vision, researchers use various augmentation methods, including cropping, color distortion, and Gaussian blurs, to generate image views [14]. However, due to graph structure's discrete nature, constructing informative views of a graph is a challenging task. Most existing works [5], [11], [12] utilize nodes as views for contrastive learning, which lacks the ability to guide GNNs to capture the global information of graphs during pre-training, and thus limits the performance enhancement during fine-tuning on downstream tasks. On the other hand, even though subgraphs are higher-level views superior to nodes for capturing global information, sampling informative subgraphs for contrastive learning is a non-trivial task. Existing sampling techniques such as random walk and k-hop neighbors are non-ideal because they only consider local structures and overlook node features, resulting in node chains or rings, which are not semantically meaningful a subgraph.

To study the characteristics of meaningful subgraphs, researchers in the graph mining community have proposed to uncover global properties of graphs through graph motifs, which are defined as significant subgraph patterns that frequently occur [15]. For example, hydroxide ($-OH$), a functional group, usually implies higher water solubility of small molecules, and $Zif268$, a protein structure, can mediate protein-protein interactions in sequence-specific DNA-binding proteins. [16]. Due to motifs' expressiveness, we propose to learn motifs from a given graph dataset and leverage learned motifs to sample informative subgraphs for GNN contrastive learning. However, existing motif mining techniques cannot be used directly for our purpose because they rely on discrete counting of subgraph structures [15], [17], [18]. This limitation makes it hard to generalize to large graph datasets with continuous and high-dimensional node features, as is often the case in real applications.

In light of the significance and challenges of motif learning, we propose *MICRO-Graph:* a framework for MotIf-driven Contrastive leaRning Of Graph representations to: 1) use GNNs to automatically extract graph motifs as prototypical subgraph embeddings from large graph datasets; 2) leverage the learned

Manuscript received 25 February 2023; revised 2 December 2023; accepted 30 January 2024. Date of publication 8 February 2024; date of current version 12 July 2024. This work was supported in part by NSF under Grant 2211557, Grant 1937599, and Grant 2119643, in part by NASA, in part by SRC, in part by Okawa Foundation Grant, in part by Amazon Research Awards, in part by Cisco Research Grant, in part by Picsart Gifts, and in part by Snapchat Gifts. Recommended for acceptance by Z. Guan. *(Shichang Zhang and Ziniu Hu contributed equally to this work.) (Corresponding author: Shichang Zhang.)*

The authors are with the Department of Computer Science, University of California, Los Angeles, CA 90095 USA (e-mail: shichang@cs.ucla.edu; bull@cs.ucla.edu; arjunsub@cs.ucla.edu; yzsun@cs.ucla.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2024.3364059, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2024.3364059

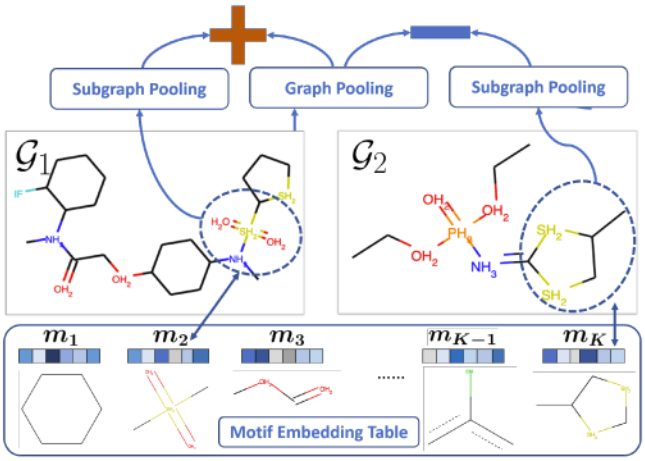


Fig. 1. Given a graph dataset, we learn a motif embedding table storing prototypical embeddings of motifs. For a pair of input graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, we leverage learned motifs to generate motif-like subgraphs and conduct graph-to-subgraph contrastive learning.

graph motifs to generate informative motif-like subgraphs to benefit contrastive learning of GNNs. The motif learning module and the contrastive learning module are mutually reinforced to train an expressive GNN that can extract meaningful motifs. Fig. 1 illustrates of this idea.

For motif learning, we tackle the challenging discrete graph motif mining problem by representing motifs as continuous embeddings, so the framework becomes differentiable. Specifically, we encode sampled motif-like subgraphs via a GNN encoder and get subgraph embeddings. Then, we treat graph motifs as a latent variable and learn them by maximizing the graph likelihood through an EM algorithm.

For contrastive learning, we tackle the challenge of informative subgraph generation by leveraging learned motifs as guidance to generate motif-like subgraphs. Specifically, we partition all nodes in a graph and induce subgraphs with a high probability of belonging to a specific motif. As motifs represent the critical graph properties by nature, the motif-like subgraphs are more informative than graph nodes and randomly sampled subgraphs, thus enhancing the contrastive learning to capture global graph characteristics.

The pre-trained GNN using *MICRO-Graph* on the ogbg-molhiv molecule dataset can successfully learn meaningful functional groups as motifs, including Benzene rings, nitro, acetate, etc., which help interpret the model decisions. Meanwhile, fine-tuning this GNN on chemical property prediction benchmarks yields 2.04% ROC-AUC average improvement over non-pretrained GNNs and outperforms other state-of-the-art GNN pre-training baselines. Extensive ablation studies show the important role of motif learning.

We summarize the contributions of this paper as follows:

- Utilize graph motifs to generate more informative subgraphs to improve contrastive GNN pre-training.
- Turn the discrete and non-scalable motif learning problem into differentiable so that we can extract significant motifs from large graph datasets with rich features.
- Achieve the best results on various chemical property prediction tasks than existing GNN pre-training techniques, and the learned motifs can facilitate researchers to interpret model decisions and scientific discovery.

II. Related Work

Contrastive learning: is a widely-used SSL algorithm, which achieves great results for visual representation learning [14], [19], [20]. One key component in contrastive learning is to generate informative and diverse views from each data instance. In computer vision, researchers leveraged a pixel as a local view to conduct local-to-local [21] or local-to-global contrastive learning [22], [23], while recently, they have found that randomly-cropped image snippets [14], [19] help a model to better capture the relationships between image elements. This motivates us to conduct contrastive learning of GNNs at the subgraph level.

Contrastive learning for GNNs: has drawn much attention recently. Existing works construct different contrastive views to pre-train GNNs [5], [6], [11], [12], [13] For example, DGI [5] performs node-to-graph contrastive learning to maximize the mutual information between node representations and the pooled global graph representation. GCC [7] conducts subgraph-to-subgraph structure contrastive learning with contrastive views generated by random walk sampling. Similar to the limitation of using a pixel as a local view, GNNs trained via node-level contrastive learning can only utilize local graph structure to determine whether a node belongs to a graph, but they are less effective at capturing whole-graph characteristics. On the other hand, the subgraph-based contrastive learning methods often rely on heuristic strategies like random walk [7], [13]. However, as these heuristic sampling strategies are random and only consider graph structures but not features, the sampled subgraphs are likely non-meaningful. For example, using random walks on molecules is likely to generate a chain graph, which is less helpful for contrastive learning. To tackle this limitation, we propose to leverage graph motifs, i.e., significant and frequently-occurring subgraph patterns, to guide informative subgraph generation for contrastive learning.

Motif-based GNN pre-training: Grover [24] proposes to use motifs as self-supervision. The main difference is that they utilize traditional software to extract discrete motifs and treat them as node classification labels. Our work instead learns motifs through joint pre-training, and we leverage the motifs, which could encode richer graph semantics, to benefit contrastive learning. MGSSL [25] performs motif-based graph SSL. MGSSL is similar to our work as both cover GNN SSL and leverage motifs, but there are three major differences. First, MGSSL conducts generative instead of contrastive pre-training, and the pre-training loss is to predict the topology of a constructed motif tree and predict masked node/edge attributes. Second, MGSSL utilizes domain knowledge and the BRISC algorithm [26] to generate motifs as a preprocessing procedure. The motif vocabulary in MGSSL is thus pre-built, whereas our motifs are learned by the GNN. Third, MGSSL can only be applied to molecules but not other graphs. We will show that our method can be more

generally applied to biology graphs. SAN [27] is a novel GNN model with a Substructure Assembling Unit (SAU) to hierarchically assemble useful pieces of graph components to fabricate discriminative substructures (motifs). To better discover motifs, SAN introduces a Soft Sequence with Context Attention (SSCA) module to deal with sampled unordered neighbor sequences. SAN is similar to this work as both try to leverage motifs. The difference is that we focus on SSL whereas SAN focuses on supervised learning and GNN model design. PASCAL [28] performs contrastive learning by generating subgraph views based on motifs. These views can capture richer information than node-level views and are structure-aware. The general idea is similar to our work, but a difference is that our motifs are learned as an intermediate step of the contrastive learning framework, whereas the motifs in PASCAL are predefined, which are only structure-based and do not consider node feature information.

Molecular graph pre-training: is another related research topic. PanGu [29] pre-trains an asymmetric graph-to-sequence conditional VAE to generate Self-Referencing Embedded Strings (SELFIES). MolGPT [30] performs a generative pre-training (GPT) of a transformer decoder on the SMILES strings. Mole-BERT [31] introduces a pre-training framework with Masked Atoms Modeling (MAM) and triplet masked contrastive learning (TMCL). MAM pre-trains GNNs by predicting masked discrete code representations of atom attributes. TMCL models the heterogeneous semantic similarity between molecules for effective molecule retrieval. MoCL [32] leverages domain knowledge at local and global levels to aid representation learning of molecular graphs. The local-level knowledge guides the augmentation process while preserving graph semantics, and the global-level knowledge captures similarity between graphs to learn richer semantics. MPG [33] performs GNN pre-training on molecules via replaced components detection, where each molecule is split into sub-parts then shuffled and combined. The model is trained to detect whether the combined parts belong to the same molecule. D-SLA [34] conducts discrepancy-based SSL by creating multiple perturbations of the given graph with varying degrees of similarity, and trains the model to predict whether a graph is perturbed. Denoising [35] pre-trains the encoder by predicting the noise added to atomic coordinates of the 3D geometry of molecule graphs. Similarly, GeoSSL [36] pre-trains the encoder via distance denoising to model the dynamic nature of the 3D geometry of molecule graphs. MEMO [37] Performs multi-view contrastive learning by combining various types of features including SMILES strings, 2D graphs, 3D geometry, and molecule fingerprints. A comprehensive survey of other methods can be found in [38]. Finally, [39] is an extended abstract of this paper published at AAAI 2021 Undergraduate Consortium, which covers the main idea of motif-driven contrastive learning.

III. Method

A. Motif-Driven Contrastive Learning Framework

The goal is to pre-train a GNN encoder $\mathbf{ENC}_\theta(\cdot)$ to capture significant characteristics of graphs using SSL, where θ denotes the parameters of the encoder. The pre-trained GNN can encode graphs in the same domain to d-dimensional embeddings capturing their fundamental semantic properties. They can be generalized to various downstream graph tasks even with few labeled data for fine-tuning.

To guide the GNN to capture global graph characteristics, we use subgraph-level contrastive learning as the self-supervised objective. The major challenge is to sample semantically-informative subgraphs. We propose to learn graph motifs via a differentiable update and leverage the learned motifs to sample informative subgraphs to tackle it. Formally, we represent motifs in a continuous embedding space as a K-slot Motif Embedding Table $M = \{m_1, \ldots, m_K\}$. Each slot stores a continuous d-dimensional motif embedding, which is a prototypical representation of subgraph embeddings. Guided by the learned motif embeddings M, we group nodes to form subgraphs with a high probability of belonging to a specific motif. These motif-like subgraphs are informative and can benefit contrastive learning.

To learn both the GNN encoder $\mathbf{ENC}_\theta$ and the Motif Embedding Table M jointly, we propose a differentiable learning framework *MICRO-Graph*, as is illustrated in Fig. 2. We first pass a batch of graphs into the GNN encoder $\mathbf{ENC}_\theta$ to get their contextualized node embeddings. Then, we group nodes to sample motif-like subgraphs, which are further pooled to get subgraph embeddings. Afterward, we feed the subgraph embeddings into the two learning modules. The *Motif-Learner* updates the motif embeddings M by maximizing the likelihood of these subgraphs, and the *Contrastive Learning* module updates GNN parameters θ. We introduce each module in detail in the following sections.

B. Differentiable Motif Learning

We first introduce how to automatically learn motif embeddings M, as well as leverage them to generate informative motif-like subgraphs in a fully-differentiable manner. This contains two coupled optimization problems: (1) given motif embeddings M, how do we partition a graph into a set of subgraphs that are most similar to the prototypical motifs; (2) given the set of subgraphs, how can we update M and get better prototypical motifs?

We derive the probabilistic model by drawing an analogy to the standard topic model, i.e., PLSA, where motif correspond to topics. Given a graph $\mathcal{G}$, we denote a partition of its N nodes $\{u_1, \ldots, u_N\}$ as $\boldsymbol{Par}$, and $\boldsymbol{Par}$ groups nodes in $\mathcal{G}$ into a set of subgraphs, i.e. $\mathcal{G}[\boldsymbol{Par}] = \{g_1, \ldots, g_J\}$. In order to model the likelihood of generating these subgraphs from motifs, we define a K-way categorical random variable $z_j \in \{1, \ldots, K\}$ with $P(g_j|z_j = k)$ representing the probability of g_j generated from the k-th motif. Hence, each subgraph can be generated with probability:

$$P(g_j|M, \theta) = \sum_{k=1}^{K} P(g_j|z_j = k, M, \theta) P(z_j = k) \quad (1)$$

For simplicity, we assume the prior probability of any subgraph belongs to motif k is uniform, i.e., $P(z_j = k) = \frac{1}{K}$.

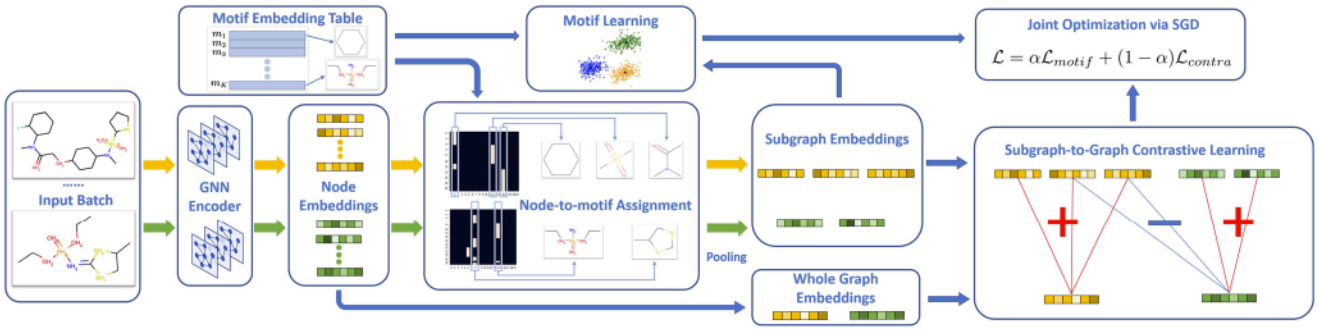


Fig. 2. Overall framework of *MICRO-Graph*. A GNN trained with SSL to automatically extract motifs. The learned motifs are leveraged to generate informative subgraphs for graph-to-subgraph contrastive learning.

We then write down the conditional likelihood of the whole graph $\mathcal{G}$ given a partition $\boldsymbol{Par}$ as:

$$P(\mathcal{G}|\boldsymbol{Par}, M, \theta) = \prod_{g_j \in \boldsymbol{Par}} \sum_{k=1}^{K} P(g_j|z_j = k)P(z_j = k) \tag{2}$$

Our first learning objective is to update the motif embedding M to maximize the conditional likelihood in (2). This is a standard clustering problem involving hidden variables z and can be optimized via an EM algorithm.

The next problem is to find the optimal partition $\boldsymbol{Par}$ that maximizes the complete likelihood $P(\mathcal{G})$ as the marginalization over all possible partitions. This task is challenging as the partition $\boldsymbol{Par}$ is a combinatorial set and is hard to represent as a latent variable. Furthermore, it is computationally intensive to enumerate all possible subgraphs and calculate $P(g_j|z_j = k)$, as the space grows exponentially in the graph size. To alleviate this issue, we propose to approximate $P(g_j|z_j = k)$ by breaking it down to the node level. Specifically, we define another K-way categorical random variable c_l with $P(u_l|c_l = k)$ representing the probability of node u_l being generated from the k-th motif, which is analogous to the word distribution for each topic.

Assuming nodes within a subgraph are mutually independent, we derive an approximated conditional likelihood:

$$\begin{aligned}&\hat{P}(\mathcal{G}|\boldsymbol{Par}, M, \theta)\\ &= \prod_{g_j \in \boldsymbol{Par}} \sum_{k=1}^{K} \prod_{\substack{l=1 \\ u_l \in g_j}}^{N} P(u_l|c_l = k)P(z_j = k)\end{aligned} \tag{3}$$

In this way, if we can estimate the node-to-motif assignment probability $P(c_l = k|u_l) \propto P(u_l|c_l = k)$, searching for the optimal partition simply becomes a dynamic programming problem that can be solved in $O(NK)$ time (maximizing the total product of values in a $N \times K$ value matrix). By further discretizing $P(c_l = k|u_l)$, partition searching can be achieved in linear time.

To summarize, motif learning involves solving two coupled optimization problems sequentially, where we based on the current M to optimize over partitions, and then optimize M given the the optimal partition we found. Note that the first objective doesn't optimize the motif embeddings M, as the node-to-motif probability is an approximation. The second objective only optimizes the motif embeddings M.

$$\boldsymbol{Par}^*,\ \theta^* = \arg\max_{\boldsymbol{Par}, \theta} \hat{P}(\mathcal{G}|\boldsymbol{Par}, M, \theta) \tag{4}$$

$$M^* = \arg\max_{\boldsymbol{M}} P(\mathcal{G}|\boldsymbol{Par}^*, M, \theta) \tag{5}$$

Modeling and Learning for Graph Partition: We first introduce how we model and optimize the graph partition problem in (4) via an EM algorithm. We denote the probability $P(c_l = k|u_l)$ as $\boldsymbol{q}_{l,k}$. We model it as the similarity between node and the motif in the embedding space. We first generate node embedding h_l of each u_l via the GNN encoder, i.e. $\{h_1, \ldots, h_N\} = \mathbf{ENC}_\theta(\mathcal{G})$. After that we project $h_l^{(i)}$ with a projection parameter matrix W_h to map the node embeddings into the motif embedding space. We model $\boldsymbol{q}_{l,k}$ as the pairwise cosine similarities between the projected node embeddings and motif embeddings, followed by a softmax normalization to turn the values into probabilities.

$$\boldsymbol{q}_{l,k} = \frac{\exp\left(\phi(W_h \boldsymbol{h}_l)^T \phi(\boldsymbol{m}_k)/\tau\right)}{\sum_{l'} \exp\left(\phi(W_h \boldsymbol{h}_{l'})^T \phi(\boldsymbol{m}_k)/\tau\right)} \tag{6}$$

Here we use $\phi(x) = x \,/\, \|x\|_2$ to denote L-2 normalization, and τ is a temperature hyper-parameter. Given a batch of graphs $\mathbb{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_B\}$, we use $\boldsymbol{Q} = [\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(B)}]^T$ to denote the corresponding node-to-motif probabilities for all N_B nodes in B graphs.

In the E-step, we would like to calculate the cluster assignment $P(c_l = k|u_l)$. However, directly calculate it via (6) could be problematic, as there exists a degenerate solution in which all embeddings collapse together and get assigned to the same motif. Similar issues were observed in previous works when researchers were performing representation learning and assignment estimation together [40], [41], [49]. To avoid this issue, we adopt the strategy used in [40] to derive $\boldsymbol{Q}^*$ as an optimal estimate of $\boldsymbol{Q}$ by solving a regularized optimization problem as in (7) and discretize it by only keeping the maximum column [40]:

$$\max_{\hat{\boldsymbol{Q}} \in \mathcal{Q}} Tr(\hat{\boldsymbol{Q}}\boldsymbol{Q}^T) + \frac{1}{\lambda} H(\hat{\boldsymbol{Q}}), \text{ where} \tag{7}$$

$$\mathcal{Q} = \{\hat{\boldsymbol{Q}} \in \mathbb{R}_+^{N_B, K} | \hat{\boldsymbol{Q}} \mathbf{1}_{\boldsymbol{K}} = \frac{\mathbf{1}_{\boldsymbol{N}}}{N}, \hat{\boldsymbol{Q}}^T \mathbf{1}_{\boldsymbol{N}} = \frac{\mathbf{1}_{\boldsymbol{K}}}{K}\} \tag{8}$$

Here $H(\hat{Q}) = -\sum_{l,k} \hat{Q}_{l,k} \log \hat{Q}_{l,k}$ stands for entropy, 1_N and 1_K are all one vectors to force the motif assignments to be balanced. This constrained optimization problem can be solved efficiently using a fast Sinkhorn-Knopp algorithm as shown in [42].

Then we do the M-step using Q^* instead of $P(c_l = k|u_l)$, we derive the optimal partition Par^* by maximizing (3), and the objective becomes the following as $q^*_{l,k}$ is either zero or one.

$$Par^* = \arg\max_{Par} \prod_{g_j \in Par} \sum_{k=1}^{K} \prod_{\substack{l=1 \\ u_l \in g_j}}^{N} q^*_{l,k} \tag{9}$$

If a subgraph contains nodes with different motif assignments, the product will be zero. Therefore, one optimal solution is simply grouping all the nodes belonging to the same motif as a subgraph. Following this simple rule, (9) is actually a straightforward step, whereas (7) do most of the heavy lifting. One concern of this simple procedure is that some learned motifs may contain too many nodes. We thus add the balanced-assignment regularization, this solution can nicely partition the graph into multiple subgraphs that have a high probability of belonging to a specific motif, which matches our design purpose. We also add randomness to this derived partition. For each subgraph, we randomly remove 10% nodes. These procedures allow us to get diverse partitions and avoid bloated motifs.

Additionally, we update the GNN parameters θ to maximize the likelihood of generating the optimal partition. Given the calculated assignment Q^* in the E-step, the negative log-likelihood loss for updating θ is the following:

$$\mathcal{L}_{node-mot} = -\sum_{l=1}^{N} q^*_l \cdot \log q_l \tag{10}$$

One potential improvement of this partition method is to add another regularization term to include graph structural information explicitly. Since in the current setting we don't constrain the nodes in sampled subgraphs to be connected, and the graph structure information is only used implicitly by generating node embeddings using GNN. This could result in an assignment that relies too much on feature information but overlooks structural information. To capture structural information more explicitly, we include another, spectral clustering-based regularization term $\mathcal{L}_{reg}$ proposed by [43]:

$$\mathcal{L}_{reg} = -\frac{Tr(q^T A q)}{Tr(q^T D q)} + \left\| \frac{\tilde{q}^T \tilde{q}}{\|\tilde{q}^T \tilde{q}\|_F} - \frac{I_J}{\sqrt{J}} \right\|_F \tag{11}$$

where $||\cdot||_F$ denotes the Frobenius norm, A and D are the adjacency and degree matrix of $\mathcal{G}$, and $\tilde{q}$ is q with only J selected columns, i.e. only the motifs we are producing corresponding subgraphs. This regularization guides GNN to make $\hat{Q}$ closer the result of spectral clustering, and penalizes disconnected nodes being assigned to the same motif, thereby balancing the usage of feature and structural information in motif-like subgraph generation. We experiment and discuss more about this extra term in the ablation study Section IV-E.

Modelling and Learning for Motif Embedding: We now introduce how given the optimal partition we model and optimize the motif embeddings in (5). After we get the partition Par^* with J subgraphs, we group and pool the node embeddings to get subgraph embeddings $\{s_j\}_{j=1}^{J}$. Similar to the node-to-motif assignment, we estimate the probability $P(z_j = k|g_j) = p_{j,k}$ as the normalized similarity between subgraph embeddings projected via W_s and the motif embeddings:

$$p_{j,k} = \frac{\exp(\phi(W_s s_j)^T \phi(m_k)/\tau)}{\sum_{j'} \exp(\phi(W_s s_{j'})^T \phi(m_k)/\tau)} \tag{12}$$

As we mentioned above, the objective in (12) now becomes a standard clustering problem like the pLSA topic model. We thus solve it with another standard EM algorithm. For E-step of calculating the assignment $\pi_{j,k}$, we directly infer that with our result Q^* from the node level, because subgraph sampled by our partition operation only contains nodes belonging to the same motif. We thus set $\pi_{j,k} = Q^*_{l,k}, \forall u_l \in g_j$. Based on this, we update the motif embeddings M by maximizing the data likelihood as the following:

$$\mathcal{L}_{mot-sub} = -\sum_{j=1}^{J} \pi_j \cdot \log p_j \tag{13}$$

As all the modules that get the loss are differentiable, we can accumulate all the losses together as (14) and use a gradient-based optimizer to update both the GNN parameters θ and motif embeddings M jointly.

$$\mathcal{L}_{motif} = \lambda_n \mathcal{L}_{node-mot} + \lambda_s \mathcal{L}_{mot-sub} + \lambda_r \mathcal{L}_{reg} \tag{14}$$

Algorithm 1: PyTorch code, Full Version in Appendix B, available online.

```
enc = GNN(args) # Any GNN Model could be used
M = nn.Parameters(K, embed_dim) # Motif Table
def forward(data)
    h, e = enc(data) #node and graph embeddings
    # Get Motif-like Subgraphs via Partition
    Q = cos_sim(W_h*h, M.detach()).softmax()
    with torch.no_grad():
        # Don't store gradient for discrete ops
        Q_hat = sinkhorn(Q)
        s, P_hat, num_subs = pool_sub(h, Q_hat)
    # Calculate the two loss for joint learning
    l_m = motif_loss(Q, Q_hat, data.adj, s, P_hat)
    l_c = contrastive_loss(s, e, num_subs)
    return l_m + l_c
def motif_loss(Q, Q_hat, adj, s, P_hat):
    # Calculate motif-to-subgraph score
    P = cos_sim(W_s*s.detach(), M).softmax()
    # Calculate the two loss via M-step
    loss_mot_sub  = -(P_hat * P.log()).mean()
    loss_node_mot = -(Q_hat * Q.log()).mean()
    loss_reg      = spectral_loss(Q, adj)
    return loss_mot_sub + loss_node_mot + loss_reg
def contrastive_loss(s, e, num_subs):
    # Force pairs from the same graph closer
    Y_lab = block_diag(num_subs)
    Y = cos_sim(W_e * e, s).softmax()
    return -(Y_lab * Y.log()).mean()
```

C. Motif-Guided Contrastive Learning

Based on the informative sampled motif-like subgraphs, we can train the GNN encoder $\mathbf{ENC}_\theta(\cdot)$ via contrastive learning on the subgraph level. Specifically, we denote the graph embeddings of $\mathbb{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_B\}$ as $\{e_1, \ldots, e_B\}$. For all $J_B = \sum_{i=1}^{B} J_i$ subgraphs sampled from this batch, we compute

the graph-to-subgraph similarity matrix $\boldsymbol{Y} \in \mathbb{R}^{B \times J_B}$, where the entry (i, j) is the cosine similarity between the corresponding graph-subgraph pair, i.e. $\boldsymbol{Y}_{i,j} = \phi(W_e e_i)^T \phi(s_j)$, where W_e is a projection parameter matrix. For each graph $\mathcal{G}_i$, subgraphs sampled from it are considered as positive pairs to it, while subgraphs from other graphs are considered as negative pairs. Thus, the contrastive objective function is calculated as follows:

$$\mathcal{L}_{contra} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{g_j \in \mathcal{G}_i} \log \frac{\exp(\boldsymbol{Y}_{i,j}/\tau)}{\sum_{j'} \exp(\boldsymbol{Y}_{i,j'}/\tau)} \tag{15}$$

D. MICRO-Graph Joint Training

The whole framework of *MICRO-Graph* can be trained jointly with a weighted sum of the two loss described above.

$$\mathcal{L} = \alpha \mathcal{L}_{motif} + (1-\alpha)\mathcal{L}_{contra} \tag{16}$$

The training steps are shown in Algorithm 1, where the motif learning and contrastive learning modules mutually enhance each other. Initially, motif embeddings are randomly initialized, and the motif-like subgraphs are random. As training proceeds, GNN can generate more representative subgraph embeddings for motif learning, while more informative subgraphs can benefit contrastive learning.

E. Time Complexity Analysis

We analyze the time complexity of our method and compare it with other representative methods in the literature. As a batch may involve graphs with different sizes, we analyze the time complexity for a batch of B input graphs, in terms of the average number of graph nodes N, the average number of graph edges M, the total number of motifs K, the average number of motifs (or views for baseline methods) selected from each graph k, and the batch size B.

Our model contains three major modules, the GNN encoder, the motif learning module, and the final contrastive learning module. The GNN encoder and contrastive module are standard modules and are used in most of the graph contrastive learning works. As shown in [12], the time complexity of these parts is linear in the number of edges, i.e., $O(BM)$. Our special part that is different from other works is the motif learning module. This module has three operations, computing $\boldsymbol{Q}$, the Sinkhorn algorithm for computing $\boldsymbol{Q}^*$, and computing $\mathcal{L}_{reg}$. The time complexity for computing $\boldsymbol{Q}$ is $O(BNK)$. [42] shows that the Sinkhorn algorithm has complexity $O(BNK)$. [43] shows that the computation of $\mathcal{L}_{reg}$ is dominated by the numerator of the first term, which is $O(N^2k + Nk^2)$ for each graph. Put these together and simplify, we get the final complexity $O(BNK + BN^2k)$. As a comparison, common graph augmentation methods considered by previous works can involve random walk sampling, e.g., GCC, GraphCL, etc, which has complexity $O(BNrk)$, with r denoting the walk length. Therefore, when K and rk are comparable, or N and r are comparable, our method has comparable complexity to these random-walk-based methods. In practice, this is often the case for molecule graphs and biological protein graphs. Another example is sampling by spectral clustering, which has time complexity $O(BN^3)$ because of the Laplacian matrix computation, and it can be much slower than our method.

IV. Experiments

We evaluate the effectiveness of *MICRO-Graph* from two perspectives: 1) whether the self-supervised framework can learn better GNNs that generalize well on downstream graph classification tasks; 2) whether the learned motifs are reasonable and can genuinely benefit contrastive learning.

A. Dataset

We conduct experiments on different types of data, including molecule graphs, biological graphs, and social graphs. For molecule graphs, we focus on chemical property prediction tasks, where large-scale unlabelled molecules are available, and many downstream tasks are label-scarce. Specifically, we pre-train GNNs using *MICRO-Graph* on the ogbg-molhiv dataset from Open Graph Benchmark (OGB) [44], which contains 40K molecules. We test our pre-trained model on smaller ogbg molecule property prediction benchmarks. For a detailed description of the OGB datasets, please see Appendix G, available online. We also consider four datasets that have been frequently used by previous graph self-supervised learning papers [7], [11], [13]. The NCI1 and DD biological graph datasets and the RDT-B and IMDB-B social network datasets. NCI contains of compounds screened for ability to suppress or inhibit the growth of a panel of human tumor cell lines. DD contains graphs with protein structures that can be classified into enzymes or non-enzyme. RDT-B contains graphs corresponding to an online discussion thread on Reddit. IMDB-B contains graphs with actor/actress and genre information of different movies.

B. Baselines and Model Configuration

We consider eight GNN SSL methods as baselines. Note that some methods had different experiment settings as this work, e.g., a different pre-train dataset. Therefore, the performance results on the same downstream task dataset can be slightly different from those original papers.

InfoGraph: [11] maximizes the mutual information between the representations of the whole graphs and the representations of its substructures.

Context prediction: [12] predicts the surrounding structure of each node, so nodes appearing in similar structural contexts will be mapped to nearby representations.

GPT-GNN: [6] predicts masked edges and node attributes. The edge prediction makes node representations to be close when there are edges between them.

GROVER: [24] first uses professional software, e.g. RDKit [45], to extract functional groups (motifs) from a dataset. Then, it pretrains by predicting motif labels.

GraphCL: [13] performs contrastive learning with four types of view augmentations: node dropping, edge perturbation, attribute masking, and subgraph sampling. In our experiments, we adopt the default setting, i.e., randomly choose two out of four methods to construct views.

TABLE I
TRANSFER FINE-TUNE PERFORMANCE (ROC-AUC) OF *MICRO-GRAPH* COMPARED WITH OTHER SELF-SUPERVISED LEARNING (SSL) BASELINES ON MOLECULE PROPERTY PREDICTION BENCHMARKS

SSL methods	bace	bbbp	clintox	hiv	sider	tox21	toxcast	Average
Non-Pretrain	72.80 ± 2.12	82.13 ± 1.69	74.98 ± 3.59	73.38 ± 0.92	55.65 ± 1.35	76.10 ± 0.58	63.34 ± 0.75	71.19
ContextPred	73.02 ± 2.59	80.94 ± 2.55	74.57 ± 3.05	73.85 ± 1.38	54.15 ± 1.54	74.85 ± 1.28	63.19 ± 0.94	70.65 (-0.54)
InfoGraph	76.09 ± 1.63	80.38 ± 1.19	**78.36 ± 4.04**	72.59 ± 0.97	**56.88 ± 1.80**	76.12 ± 1.11	64.40 ± 0.84	72.11 (+0.93)
GPT-GNN	75.56 ± 2.49	83.35 ± 1.70	74.84 ± 3.45	74.82 ± 0.99	55.59 ± 1.58	76.34 ± 0.68	64.76 ± 0.62	72.18 (+0.99)
GROVER*	75.22 ± 2.26	83.16 ± 1.44	76.8 ± 3.29	74.46 ± 1.06	56.63 ± 1.54	76.77 ± 0.81	64.43 ± 0.8	72.5 (+1.31)
GraphCL	76.47 ± 2.75	82.76 ± 1.00	77.74 ± 4.38	75.02 ± 1.03	56.12 ± 1.12	76.1 ± 1.34	63.25 ± 0.53	72.49 (+1.3)
DGI	76.19 ± 2.45	83.21 ± 1.75	74.08 ± 3.31	73.75 ± 1.24	55.91 ± 1.19	76.59 ± 0.61	64.17 ± 0.62	71.99 (+0.80)
GCC	74.13 ± 3.05	83.11 ± 2.15	75.12 ± 3.8	74.17 ± 1.32	56.01 ± 0.97	76.22 ± 1.14	63.39 ± 0.77	71.74 (+0.55)
MGSSL*	76.31 ± 2.15	85.21 ± 1.46	77.42 ± 4.01	75.03 ± 0.94	56.31 ± 1.27	77.01 ± 1.04	65.96 ± 0.81	73.32 (+2.13)
MICRO-Graph	**77.22 ± 2.02**	**84.38 ± 1.07**	77.02 ± 1.96	**75.07 ± 1.07**	56.67 ± 0.88	**77.04 ± 0.77**	**65.23 ± 0.82**	**73.23 (+2.04)**

Pre-train GNNs on Ogbg-Molhiv dataset, fine-tune on each downstream task for ten times. *Means the method uses domain knowledge and external tools to generate motifs.

TABLE II
FEATURE EXTRACTION PERFORMANCE (ROC-AUC) OF *MICRO-GRAPH* COMPARED WITH OTHER SELF-SUPERVISED LEARNING (SSL) BASELINES ON MOLECULE PROPERTY PREDICTION BENCHMARKS

SSL methods	bace	bbbp	clintox	hiv	sider	tox21	toxcast	Average
ContextPred	53.09 ± 0.84	55.51 ± 0.08	40.73 ± 0.02	53.31 ± 0.15	52.28 ± 0.08	35.31 ± 0.25	47.06 ± 0.06	48.18
InfoGraph	66.06 ± 0.82	75.34 ± 0.51	**75.71 ± 0.53**	61.45 ± 0.74	54.70 ± 0.24	63.95 ± 0.24	52.69 ± 0.07	64.27
GPT-GNN	59.43 ± 0.66	71.58 ± 0.54	62.78 ± 0.58	64.08 ± 0.36	54.67 ± 0.16	68.20 ± 0.14	57.06 ± 0.13	62.53
GROVER*	65.67 ± 0.38	78.47 ± 0.36	53.19 ± 0.68	69.03 ± 0.23	54.94 ± 0.12	67.63 ± 0.13	57.28 ± 0.05	63.74
GraphCL	60.93 ± 1.72	76.91 ± 0.85	73.79 ± 2.13	70.52 ± 1.22	55.01 ± 1.43	**72.14 ± 0.78**	58.51 ± 0.39	66.83
DGI	65.32 ± 1.05	74.31 ± 0.95	72.21 ± 1.08	64.73 ± 0.82	53.91 ± 0.47	62.12 ± 0.48	55.92 ± 0.67	64.07
GCC	59.21 ± 0.85	69.12 ± 0.65	63.51 ± 0.88	68.71 ± 0.15	54.12 ± 0.79	64.25 ± 0.82	56.97 ± 0.41	62.27
MGSSL*	71.23 ± 1.45	80.02 ± 0.69	74.05 ± 0.81	75.79 ± 0.54	58.82 ± 0.91	70.23 ± 0.71	60.28 ± 0.42	70.06
MICRO-Graph	**70.83 ± 2.06**	**82.97 ± 1.53**	73.49 ± 2.16	**73.34 ± 1.27**	**57.32 ± 0.62**	71.82 ± 0.82	**59.46 ± 0.25**	**69.73**

Use pre-trained models to extract graph embeddings and train linear classifiers for ten times. * means the method uses domain knowledge and external tools to generate motifs.

DGI: [5] maximizes the mutual information between local node representations and global graph representations.

GCC: [7] conducts subgraph-to-subgraph structure contrastive learning with views generated by random walk.

MGSSL: [25] conducts generative pre-training by first constructing motif trees with the BRISC algorithms [26], and then predicts the tree topology and masked attributes.

We use the state-of-the-art GNN model, Deeper Graph Convolutional Networks (DeeperGCNs) proposed in [46], as the base GNN encoder for *MICRO-Graph* and all baselines. We use the same model architecture hyperparameters recommended by the original DeeperGCN architecture for all experiments. Details about hyperparameters and model configurations are in Appendix H, available online.

TABLE III
EMBEDDING EXTRACTION EVALUATION (AVERAGE ACCURACY) OF *MICRO-GRAPH* COMPARED WITH OTHER BASELINES ON BIOLOGICAL GRAPHS (NCI1 AND DD) AND SOCIAL GRAPHS (RDT-B AND IMDB-B)

methods	NCI1	DD	RDT-B	IMDB-B
ContextPred	64.2 ± 1.5	71.4 ± 2.1	77.5 ± 1.7	72.2 ± 0.7
InfoGraph*	76.2 ± 1.1	72.9 ± 1.8	82.5 ± 1.4	73.0 ± 0.9
GPT-GNN	74.3 ± 1.6	75.8 ± 1.2	86.5 ± 1.3	70.7 ± 0.9
GraphCL*	77.9 ± 0.4	78.6 ± 0.4	**89.5 ± 0.8**	71.1 ± 0.4
DGI	78.2 ± 0.6	76.1 ± 1.2	78.6 ± 0.9	67.3 ± 1.1
GCC	70.2 ± 0.8	72.1 ± 0.4	87.6 ± 1.2	73.8 ± 1.2
MICRO-Graph	**78.7 ± 1.3**	**79.5 ± 0.7**	87.1 ± 0.5	**74.5 ± 0.7**

We report the mean and std of 5 experiments. Results with * are from previous papers under the same experiment setting. The highest results are in bold.

C. Evaluation Results Under Different Protocols

We evaluate the effectiveness of pre-trained GNNs on three types of datasets: molecule graphs, biological graphs, and social graphs. We use two evaluation protocols: fine-tune evaluation and feature extraction evaluation.

Fine-tune/Feature Extraction Evaluation on Molecule Graphs: For the molecule dataset, we evaluate by following the literature convention in [12] and mimic the real-world setting with scarce data labels. We fine-tune the pre-trained GNN model on a small labeled data portion on downstream tasks. We adopt the same train-test and model selection procedure as in [4], [47], [48], where we perform 10-fold cross-validation and report the epoch with the best cross-validation performance averaged over the 10 folds. The evaluation metric we used is the ROC-AUC score. We also compare our result to the non-pretrain (direct supervised learning) setting. The evaluation results under the transfer fine-tune setting is illustrated in Table I. Similarly, we also show a feature extraction setting where a single-layer MLP is trained on top of the extracted features. Results are shown in Table II.

Feature Extraction Evaluation on Biological and Social Graphs: For biological and social datasets, we evaluate following the literature convention in [13]. We pre-trained a GNN first and then used fix it use it only as a feature extractor to get graph representations. The generated representations are fed into a down-stream SVM classifier for evaluation. The results are shown in Table III.

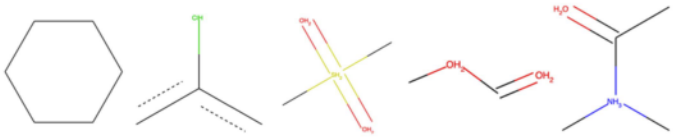


Fig. 3. Five frequent motifs learned by *MICRO-Graph* on molecular graphs, represented by their closest subgraphs.

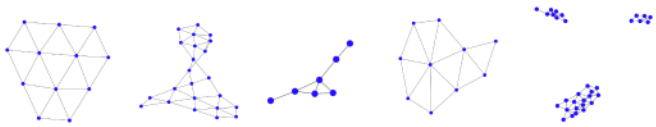


Fig. 4. Five frequent motifs learned by *MICRO-Graph* on biological graphs (DD), represented by their closest subgraphs.

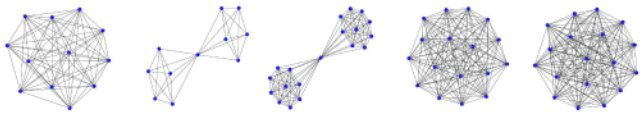


Fig. 5. Five frequent motifs learned by *MICRO-Graph* on social graphs (IMDB-B), represented by their closest subgraphs.

For both settings, *MICRO-Graph* outperforms most of the baselines on average performance and achieves the highest results on most datasets. For the transfer fine-tune, we gain about 2.04% performance enhancement against the non-pretrain baseline. The only baseline that slightly outperforms *MICRO-Graph* is MGSSL (by 0.09 on average for fine-tune evaluation, and 0.33 on average for feature extraction evaluation). However, MGSSL heavily relies on domain knowledge and only works for molecule graphs.

D. Visualization and Analysis of Learned Motifs

We show learned motifs of *MICRO-Graph* by collecting the closest subgraphs, and we observe quite different motif patterns for different graph types. For molecular graphs in Fig. 3, the learned motifs are similar to meaningful functional groups, such as Benzene rings and acetate. This shows that *MICRO-Graph* can learn reasonable and meaningful motifs. A complete list of the learned motifs is shown in Appendix C.1, available online. For biological graphs in Fig. 4, nodes represent amino acids, and edges represent their spatial proximity. We observe grid-like motifs, indicating a regular spatial arrangement of amino acids, which potentially reflect functional regions of proteins. We also observe star-like motifs, with a central amino acid connected to multiple others. This motif is connected to the protein's tertiary structure, where the central node in the star-like motif could be a highly interactive residue, such as an active site in enzymes or a binding site in receptor proteins. For social graphs in Fig. 5, we observe interesting clique motifs and motifs with one central node bridging two cliques. These patterns reveals the community structure of a social network, where there are tight-knit community where all members interact with each other and a hierarchical social structure with one popular individual connecting two groups.

TABLE IV
AVERAGE FEATURE EXTRACTION RESULT WITH DIFFERENT ABLATIONS

Ablations	Model Component	Avg.
MICRO-Graph (Default Settings)		69.73
Subgraph Samplers	Random Walk	64.09
	K-hop Sampler	64.74
	w/o $\mathcal{L}_{reg}$	67.91
Contrastive Views	Sub-Sub (MICRO-G)	65.71
	Sub-Sub (GraphCL)	66.83
	Node-Sub (InfoGraph)	64.27
Subgraph Encoding	w/o contextualized emb	61.68
Number of Motifs	K = 5	67.96
	K = 10	68.89
	K = 20 (default)	69.73
	K = 50	68.59
	K = 100	68.96
	K = 200	68.87
	K = 300	68.81

E. Ablation Study

Motif-Like Subgraph Sampling: Sampling subgraphs is a pervasive operation in graph learning. Simple approaches that often show up in literature include random walk and k-hop neighbours. The problem with these two sampling methods is that they only use local graph structural information but not feature information. Thus, they cannot accurately generate semantically-meaningful subgraphs when graph features are critical for representing the subgraph and whole-graph properties. MICRO-Graph leverages learned motifs to produce motif-like subgraphs. To evaluate its effectiveness, we run our framework by replacing motif-guided sampling with random walk and k-hop sampler, with all other settings stay the same. We show the result in the first block of Table IV. As we see, the performance of the same framework with random walk and k-hop drops 5.64% and 4.99% respectively. This shows the importance of generating motif-like subgraphs. We also conduct ablation by removing the spectral regularizer $\mathcal{L}_{reg}$, and the performance drops 1.82%. After removing this regularization, the sampled subgraphs are prone to contain disconnected nodes, which hinders the generalization performance to real-world graphs.

Two examples of subgraphs generated by all four strategies are shown in Fig. 6 . From the sampled subgraphs, we can see that random walk is more likely to generate chains (as it just randomly pick next-hop nodes, and thus have low probability to sample a complete benzene ring, which is the most basic component of molecule subgraph), while k-hop sampling is more likely to generate half part of a Benzene ring (as it just add in all the local neighborhood without selecting the important ones, and thus most subgraphs look similar). Neither of these two heuristic approaches can successfully generate a complete and clean functional group (such as benzene rings), and the generated subgraphs are not very meaningful. On the

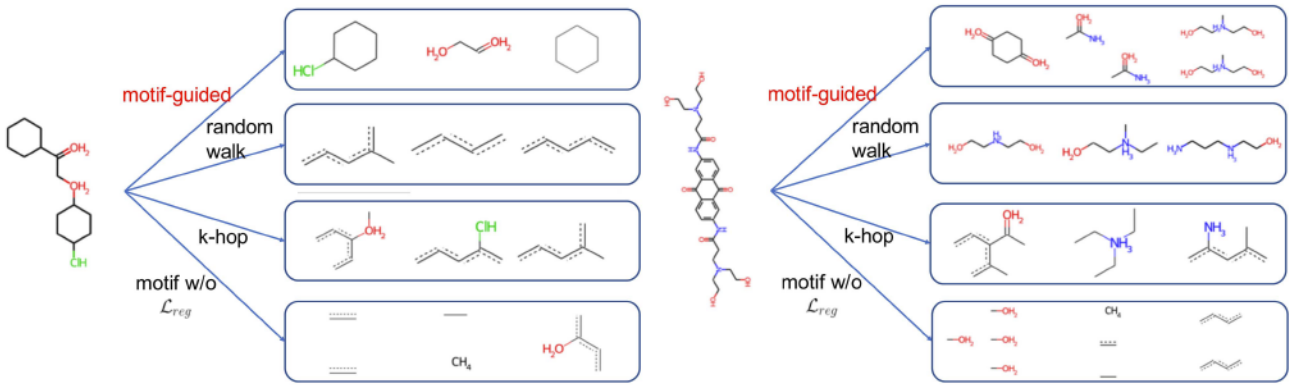


Fig. 6. Comparison between different sampling strategies. Two examples are shown. In each example, the original graph is shown on the left. Samples produced by four different sampling strategies are shown on the right. The top row shows the samples by our motif-guided segmenter.

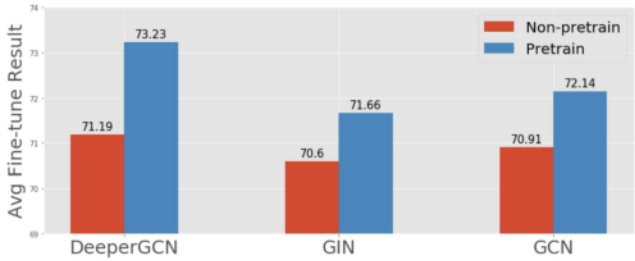


Fig. 7. Average fine-tune result over three GNN architectures.

contrary, our motif-guided sampling can successfully generate a complete benzene rings and other significant substructures. This intuitively explains why the contrastive learning with our motif-guided sampler works much better than the others. After removing the $\mathcal{L}_{reg}$ term in the joint loss, the samples can still capture important nodes in the whole graph, but it will overlook some structural information and are prone to get some disconnected parts in a subgraph, which is not very realistic.

Graph-to-Subgraph Contrastive: In this paper, we mainly utilize graph-to-subgraph (graph-sub) contrastive. Similar to image crop pair in computer vision, subgraph-to-subgraph (sub-sub) contrastive is also a promising alternative, which has already been studied in GraphCL [13] with random walk. We do an ablation study by conducting sub-sub contrastive learning utilizing our motif-like subgraphs with all other settings unchanged. Results of this setting are worse than graph-sub learning, and worse than the GraphCL baseline.

We hypothesize that the performance gap of sub-sub with motif-like subgraphs is due to the false-negative of the current graph partition procedure. By grouping nodes close to a limited number of motifs, there is a high probability that two different graphs have a similar subgraph structure, which would form a false negative pair. To overcome this limitation, it is necessary to incorporate more randomness and take the composition of motifs as subgraph, which we leave these possibilities as future directions.

One might also ask why the graph-sub setting is less sensitive to false-negatives. For our model design, one key choice is to first get all the contextualized node embeddings that encode the whole graph characteristics. In this way, even if two subgraphs from different graphs share a similar structure, their subgraph embeddings can still encode different context information. To test this hypothesis, we remove contextualized embedding by re-encoding all the sampled subgraphs. As shown in the third block, the result in this setting is extremely poor, worse than all existing pre-training frameworks. This explains the importance of contextualized node embeddings for subgraph-level contrastive learning and explains why our proposed graph-sub setting works.

Number of Motifs: The number of motif slots, K, is an essential hyperparameter in our motif learning framework. We thus conduct an ablation study with three different K values, i.e., 5, 10, 20, 50, 100, 200, and 300. As illustrated in the last block of Table IV, with different K values, *MICRO-Graph* shows consistent performance enhancement, while an intermediate value 20 gives the best result on average.

Our hypothesis is that the best K is dataset dependent, and a moderate K is optimal for molecule datasets, as this will create enough capacity for learning different motifs and does not introduce too much redundancy. We found that the results from 50 to 300 are all within reasonable variance. We conclude that as the number of motifs gets larger, model performance actually does not differ much. When there are more than enough motif slots, some slots will simply become redundant and correspond to random subgraphs. We verify this by checking the similarities between a motif embedding and its top 5 most similar subgraphs when K is large, for some motifs all these 5 values are low (cosine similarity < 0.5), which means no subgraphs are considered close to these motifs, and the motif slots are not really used. Having a bigger K creates more of these unused motif slots. On the other hand, for a very small K, e.g., $K = 5$, the performance is not good. We hypothesize the reason being too few motif slots cannot capture all meaningful substructure patterns.

GNN Architectures: The *MICRO-Graph* framework is agnostic to the GNN architecture. We show this by trying three different standard GNN architectures, i.e. GCN [1], GIN [4], and DeeperGCN [46]. As we showed in Fig. 7, *MICRO-Graph* provides a consistent performance enhancement for all different GNNs, and a more expressive GNN like DeeperGCN can provide a larger performance improvement.

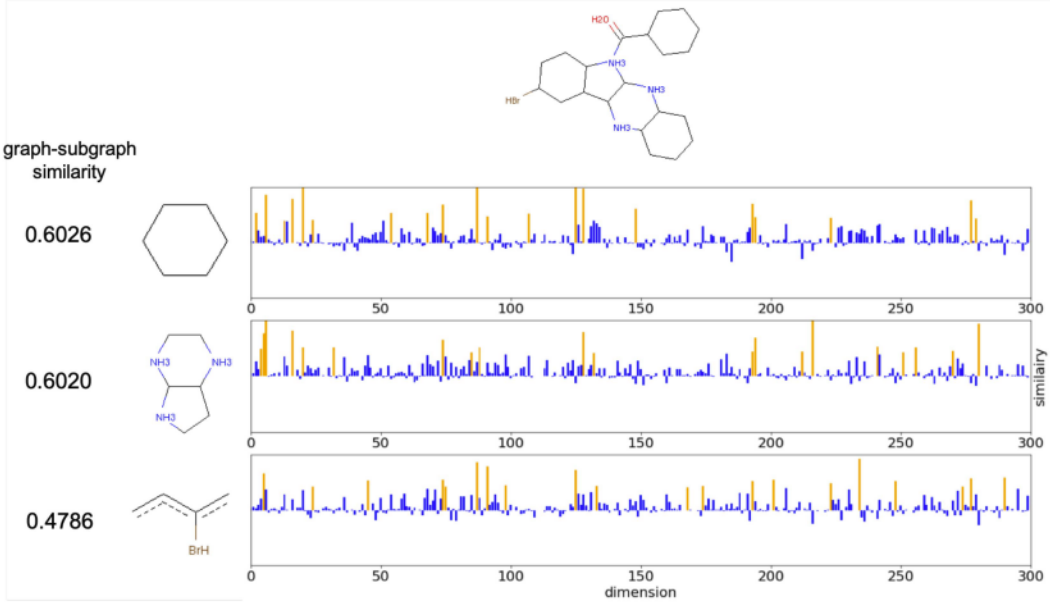


Fig. 8. Similarity between the whole graph $\mathcal{G}_1$ and three subgraphs g_1, g_2, and g_3, zoom in to each dimension. For each row, x-axis is the dimension slot 1 to 300, and y-axis is the similarity scores between corresponding dimensions of the whole graph representation and each subgraph representation. We indicate the top 20 scores in orange. We can see that these three subgraphs have very different similarity score distributions, though summing over all 300 dimensions give alike high scores.

V. Case Study of Graph-to-Subgraph Contrastive and the Learned Embeddings

Another core component in *MICRO-Graph* is the graph-to-subgraph contrastive learning based on motif-like subgraphs. Though we have previously showed that such design can empirically and intuitively help pre-traing better GNN, there's still some potential question about the combination of motif with contrastive learning.

A. Different Subgraphs From the Same Graph

One key question is that each graph can be partitioned into subgraphs that belong to different motifs. Through graph-to-subgraph contrastive learning, we force these subgraph embeddings to be similar to the whole graph embedding. However, will these subgraph embeddings also be forced to be similar to each other? If so, it contradicts to our principle of learning distinctive motif semantics.

We investigate this question by a case study of a particular graph in Fig. 8, which is partitioned into three subgraphs that belong to different motif slots. We show the similarity score of these subgraph to the whole graph, and also the entry-wise similarity of each hidden dimension of the 300-dimension embedding. As we can see, all the three subgraphs can get relatively high similarity score, compared to the overall distribution of graph-to-subgraph similarity score shown in Fig. 9.

One very interesting findings is that the maximum entry of similar score for these three subgraphs are very different. Specifically, this indicates that the 300-dimension embedding actually encode multi-view semantic information. While doing cosine similarity of subgraph to whole graph, different dimension could be activated for different subgraphs. In other words, they are

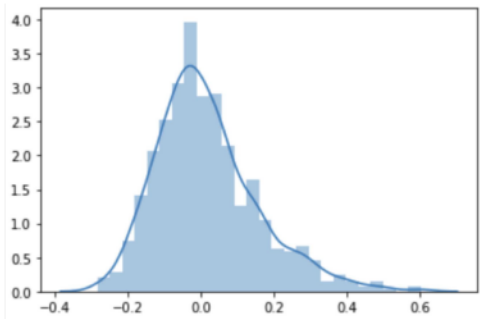


Fig. 9. Distribution of similarity scores between the whole graph G and all the subgraphs.

only similar to the projection of the whole graph representation on different basis. Therefore, even the three subgraphs are all similar to the whole graph, their embeddings do not collapse to be the same, which maintains the diversity and distinctiveness of the motifs.

To further justify our claim, we also show the pairwise cosine similarity scores between these three subgraphs in Fig. 10. We find that their mutual similarity is not very low, indicating that each subgraph embeddings could capture their own semantics. To show that this claim is generalizable to the whole dataset, we randomly select a batch of graph $\{\mathcal{G}_1, \ldots, \mathcal{G}_B\}$, and get all their subgraphs. We use Fig. 11 to show the pairwise similarity scores between these subgraphs in order. we find that even though these subgraphs are listed in order, (i.e. $g_1, \ldots, g_3$ are from $\mathcal{G}_1$, g_4, g_5 are from $\mathcal{G}_2$, $g_6, \ldots g_8$ are from $\mathcal{G}_3$, and etc) similarity scores are roughly uniform. In other words, this heat matrix is not strictly block diagonal, indicating subgraphs from the same whole graph

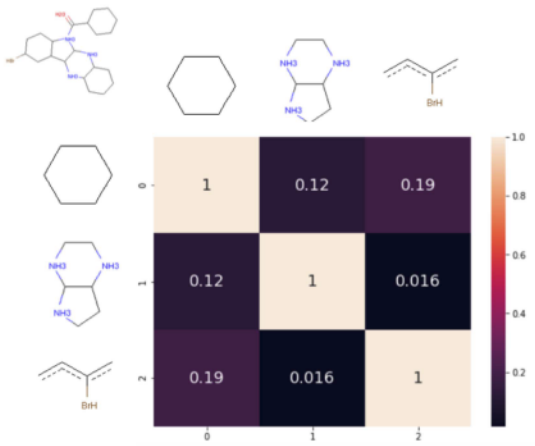


Fig. 10. Pairwise similarity scores between subgraphs g_1, g_2, and g_3 from the same whole graph.

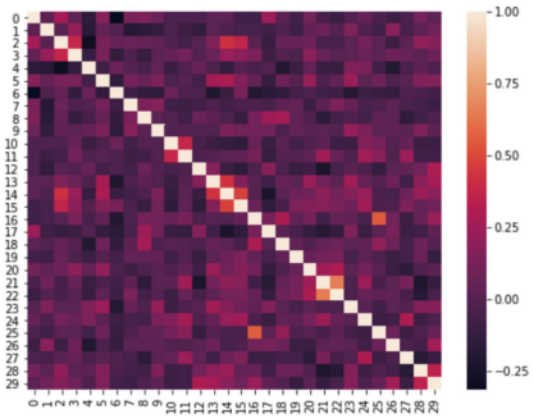


Fig. 11. Pairwise similarity scores between the first 30 subgraphs sampled from $\{\mathcal{G}_1, \ldots, \mathcal{G}_B\}$.

do not necessarily have high similarities among them. This again justifies that our claim is true among the whole dataset.

Also, this study partially answers why graph-to-subgraph contrastive works better than subgraph-to-subgraph contrastive in our setting. Using subgraph-to-subgraph contrastive, two subgraph in the same graph that have different motif assignments will be forced to be similar, which contradicts to our assumption to motif embedding. If we want to extend the current framework to subgraph-to-subgraph contrastive, a better subgraph sampling that considers composition of motif and more randomness is required.

B. Performance Attribution: Contrastive Versus Generative

Methods covered in the experiment section fall into two categories, contrastive and generative, where the best performance was achieved by the generative-based method, MGSSL, which also uses predefined motifs. We attribute the performance gain of MGSSL mostly to the predefined motifs. The conclusion is drawn from comparing GROVER and GPT-GNN. GROVER relies on predefined motifs for pre-training, where it extracts motifs using professional molecular software and then pre-trains by predicting motif occurrences in molecule graphs. In contrast, GPT-GNN employs a generative SSL approach to generate masked nodes, edges, and node features following a specific order, without utilizing additional domain knowledge like predefined motifs. Despite the relative simplicity of GROVER's pre-training objective, which resembles supervised learning with predefined motifs as labels, it outperforms GPT-GNN (72.5 vs. 72.18). For MGSSL, its strong performance is primarily rooted in the utilization of predefined motifs, as its generative SSL approach also relies on motifs rather than simple masking reconstruction of nodes, edges, and features. For MGSSL, the generation process involves constructing motif trees and the generation order is determined by either BFS or DFS on the trees. This additional complexity in MGSSL is designed to effectively leverage domain knowledge. In conclusion, for model pre-training on graph data with sufficient domain knowledge, such as molecules, finding ways to incorporate domain knowledge like predefined motifs with ML models holds promise and warrants further investigation. However, for other graph data with less domain knowledge, such as social graphs, contrastive SSL remains a preferable choice. In support of this, both our MICRO-Graph and GraphCL outperform GPT-GNN on IMDB data, whereas MGSSL and GROVER cannot be applied in this case.

VI. Conclusion

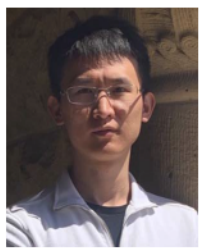
We propose *MICRO-Graph* to pre-train a GNN via subgraph-level contrastive learning. To tackle the challenge of informative subgraph sampling, we learn motifs during pre-training. With *MICRO-Graph*, we learn meaningful motifs that align with molecular functional groups. Fine-tuning the pre-trained GNN on seven chemical property prediction benchmarks yields 2.04% average improvement over non-pretrained GNNs and outperforms pre-training baselines.

Limitation and Future Work: The exploration of various downstream tasks is essential in the context of general graph contrastive learning. While our work primarily focuses on graph classification tasks, more downstream tasks like node classification are not investigated in this work. We recognize the potential of node classification as a promising avenue for future research. A straightforward way to adapt our approach to node classification is by transforming these problems into graph classification tasks, notably through the extraction of ego-graphs for individual nodes. In fact, the IMDB-B dataset explored in this work was constructed with ego-graphs, and our method shows reasonable performance for both prediction and motif learning on this dataset.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[2] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv: 1710.10903*.
[4] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv: 1810.00826*.

[5] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," 2018, *arXiv: 1809.10341*.
[6] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "GPT-GNN: Generative pre-training of graph neural networks," in *Proc. 26th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 1857–1867.
[7] J. Qiu et al., "Graph contrastive coding for graph neural network pre-training," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1150–1160.
[8] Y. Bai et al., "Unsupervised inductive graph-level representation learning via graph-graph proximity," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1988–1994.
[9] N. Navarin, D. V. Tran, and A. Sperduti, "Pre-training graph neural networks with kernels," 2018, *arXiv: 1811.06930*.
[10] L. Wang et al., "Inductive and unsupervised representation learning on graph structured objects," in *Proc. Int. Conf. Learn. Representations*, 2020s.
[11] 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*.
[12] W. Hu et al., "Strategies for pre-training graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2020.
[13] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 5812–5823, 2020.
[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.
[15] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: Simple building blocks of complex networks," *Sci.*, vol. 298, no. 5594, pp. 824–827, 2002.
[16] C. O. Pabo, E. Peisach, and R. A. Grant, "Design and selection of novel Cys2His2 zinc finger proteins," *Annu. Rev. Biochem.*, vol. 70, no. 1, pp. 313–340, 2001.
[17] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon, "Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs," *Bioinformatics*, vol. 20, no. 11, pp. 1746–1758, 2004.
[18] S. Wernicke, "Efficient detection of network motifs," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 3, no. 4, pp. 347–359, Fourth Quarter 2006.
[19] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2019, *arXiv: 1911.05722*.
[20] O. J. Hénaff et al., "Data-efficient image recognition with contrastive predictive coding," 2019, *arXiv: 1905.09272*.
[21] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv: 1807.03748*.
[22] R. D. Hjelm et al., "Learning deep representations by mutual information estimation and maximization," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.
[23] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," in *Proc. 32th Annu. Conf. Neural Inf. Process. Syst.*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., Vancouver, BC, Canada, 2019, pp. 15 509–15 519.
[24] Y. Rong et al., "Self-supervised graph transformer on large-scale molecular data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 12559–12571.
[25] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C.-K. Lee, "Motif-based graph self-supervised learning for molecular property prediction," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 15 870–15 882, 2021.
[26] J. Degen, C. Wegscheid-Gerlach, A. Zaliani, and M. Rarey, "On the art of compiling and using'drug-like'chemical fragment spaces," *ChemMedChem, Chem. Enabling Drug Discov.*, vol. 3, no. 10, pp. 1503–1507, 2008.
[27] Y. Yang, Z. Guan, W. Zhao, W. Lu, and B. Zong, "Graph substructure assembling network with soft sequence and context attention," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4894–4907, May 2023.
[28] Z. Chen, Y. Peng, S. Yu, C. Cao, and F. Xia, "Subgraph adaptive structure-aware graph contrastive learning," *Mathematics*, vol. 10, no. 17, 2022, Art. no. 3047.
[29] X. Lin et al., "PanGu drug model: Learn a molecule like a human," *Sci. China Life Sci.*, vol. 66, no. 4, pp. 879–882, Apr. 2023.
[30] V. Bagal, R. Aggarwal, P. Vinod, and U. D. Priyakumar, "MolGPT: Molecular generation using a transformer-decoder model," *J. Chem. Inf. Model.*, vol. 62, no. 9, pp. 2064–2076, 2021.
[31] J. Xia et al., "Mole-BERT: Rethinking pre-training graph neural networks for molecules," in *Proc. 11th Int. Conf. Learn. Representations*, 2022.
[32] M. Sun, J. Xing, H. Wang, B. Chen, and J. Zhou, "MoCL: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 3585–3594.
[33] P. Li et al., "An effective self-supervised framework for learning expressive molecular global representations to drug discovery," *Brief. Bioinf.*, vol. 22, no. 6, 2021, Art. no. bbab109.
[34] D. Kim, J. Baek, and S. J. Hwang, "Graph self-supervised learning with accurate discrepancy learning," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 14 085–14 098, 2022.
[35] S. Zaidi et al., "Pre-training via denoising for molecular property prediction," 2022, *arXiv:2206.00133*.
[36] S. Liu, H. Guo, and J. Tang, "Molecular geometry pretraining with SE (3)-invariant denoising distance matching," 2022, *arXiv:2206.13602*.
[37] Y. Zhu, D. Chen, Y. Du, Y. Wang, Q. Liu, and S. Wu, "Featurizations matter: A multiview contrastive learning approach to molecular pretraining," in *Proc. ICML 2nd AI Sci. Workshop*, 2022.
[38] J. Xia, Y. Zhu, Y. Du, Y. Liu, and S. Z. Li, "A systematic survey of molecular pre-trained models," in *Proc. Int. Joint Conf. Artif. Intell.*, 2023, pp. 6787–6795.
[39] A. Subramonian, "Motif-driven contrastive learning of graph representations," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 15980–15981.
[40] Y. M. Asano, C. Rupprecht, and A. Vedaldi, "Self-labelling via simultaneous clustering and representation learning," in *Proc. Int. Conf. Learn. Representations*, 2020.
[41] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2020, *arXiv: 2006.09882*.
[42] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transportation distances," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2292–2300.
[43] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 874–883.
[44] W. Hu et al., "Open graph benchmark: Datasets for machine learning on graphs," 2020, *arXiv: 2005.00687*.
[45] W. Hu et al., Open graph benchmark: Datasets for machine learning on graphs, in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 22 118–22 133.
[46] G. Li, C. Xiong, A. Thabet, and B. Ghanem, "DeeperGCN: All you need to train deeper GCNs," 2020, *arXiv: 2006.07739*.
[47] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 1365–1374.
[48] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 4438–4445.
[49] A. Van Den Oord et al., "Neural discrete representation learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6306–6315.



Shichang Zhang received the BA and MS degrees in statistics from Berkeley and Stanford respectively. He is currently working toward the forth-year PhD degree in computer science with the University of California, Los Angeles (UCLA) Data Mining Lab working with Professor Yizhou Sun. This research insterets include graph neural network, explainable AI, and Trustworthy ML.

Ziniu Hu (Member, IEEE) received the bachelor's degree in CS from Peking University, advised by Prof. Xuanzhe Liu. He is currently working toward the final-year CS PhD degree with the University of Calofornia, Los Angeles (UCLA). He had the fortune to be advised by Prof. Yizhou Sun, and worked closely with Prof. Kai-Wei Chang. His research is generously supported by Baidu PhD Fellowship and Amazon PhD Fellowship.

Arjun Subramonian is a currently working toward the PhD degree in computer science with the University of Calofornia, Los Angeles (UCLA) conducting machine learning research, working with Prof. Yizhou Sun and Prof. Kai-Wei Chang. Their research focuses on inclusive graph machine learning and natural language processing, including fairness, biases, and ethics. They cares deeply about the inclusion of LGBTQIA+ individuals and disabled and neurodivergent folks in AI research.

Yizhou Sun received the bachelor's and master's degrees in computer science and statistics from Peking University, China and the PhD degree from Computer Science Department, University of Illinois at Urbana Champaign (UIUC) in December 2012. She is currently an associate professor with Computer Science, UCLA. Prior to that, she joined Northeastern University as an assistant professor in 2013.