# MOTIF-DRIVEN CONTRASTIVE LEARNING OF GRAPH REPRESENTATIONS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Graph motifs are significant subgraph patterns occurring frequently in graphs, and they play important roles in representing the whole graph characteristics. For example, in chemical domain, functional groups are motifs that can determine molecule properties. Mining and utilizing motifs, however, is a non-trivial task for large graph datasets. Traditional motif discovery approaches mostly rely on exact counting or statistical estimation, which are hard to scale for a large number of graphs with continuous and high-dimension features. In light of the significance and challenges of motif mining, we propose *MICRO-Graph*: a framework for MotIf-driven Contrastive leaRning Of Graph representations to: 1) pre-train Graph Neural Networks (GNNs) in a self-supervised manner to automatically extract graph motifs from large graph datasets; 2) leverage learned motifs to guide the contrastive learning of graph representations, which further benefit various graph downstream tasks. Specifically, given a graph dataset, a motif learner cluster similar and significant subgraphs into corresponding motif slots. Based on the learned motifs, a motif-guided subgraph segmenter is trained to generate more informative subgraphs, which are used to conduct graph-to-subgraph contrastive learning of GNNs. By pre-training on ogbg-molhiv molecule dataset with our proposed *MICRO-Graph*, the pre-trained GNN model can enhance various chemical property prediction downstream tasks with scarce label by 2.0%, and significantly higher than other state-of-the-art self-supervised learning baselines.

## 1 INTRODUCTION

Graph-structured data, such as molecules and social networks, is ubiquitous in many scientific research areas and real-world applications. To represent graph characteristics, graph motifs were proposed in Milo et al. (2002) as significant subgraph patterns occurring frequently in graphs and uncovering graph structural principles. Functional groups are important motifs that can determine molecule properties. For example, for chemical compounds, hydroxide (–$OH$) usually implies higher water solubility, and for proteins, Zif268 can mediate protein-protein interactions in sequence-specific DNA-binding proteins. (Pabo et al., 2001).

Researchers have been studying graph motifs for years. On one hand, utilizing meaningful motifs can benefit many important applications like quantum chemistry and drug discovery (Ramsundar et al., 2019). On the other hand, extracting motifs from large graph datasets is still a challenging question. Traditional motif discovery approaches (Milo et al., 2002; Kashtan et al., 2004; Schreiber & Schwöbbermeyer, 2005; Chen et al., 2006; Wernicke, 2006) mostly rely on discrete counting or statistical estimation, and thus are hard to generalize to large-scale graph datasets with continuous and high-dimension features, as often the case in real-world applications.

Recently, Graph Neural Networks (GNNs) have shown great expressive power for learning graph representations without explicit structural feature engineering (Kipf & Welling, 2016; Hamilton et al., 2017; Veličković et al., 2017; Xu et al., 2018). In addition, GNNs can be trained in a self-supervised manner without human annotations to capture important graph structural and semantic properties (Veličković et al., 2018; Hu et al., 2020c; Qiu et al., 2020; Bai et al., 2019; Navarin et al., 2018; Wang et al., 2020; Sun et al., 2019; Hu et al., 2020b). This motivates us to rethink about motifs as more general representations than exact structure matches and ask the following research questions:

- Can we use GNNs to automatically extract graph motifs from large graph datasets?
- Can we leverage the learned graph motif to benefit self-supervised GNN learning?

In this paper, we propose *MICRO-Graph*: a framework for MotIf-driven Contrastive leaRning Of Graph representations. The key idea of this framework is to learn graph motifs as prototypical cluster centers of subgraph embeddings encoded by GNNs. In this way, the discrete counting problem is transfered to a fully-differentiable framework that can generalize to large-scale graph datasets with continuous and high-dimensional features. In addition, the learned motif can help generate more informative subgraph for graph-to-subgraph contrastive learning. The motif learning and contrastive learning are mutually reinforced, and eventually help pre-train a more generalizable GNN encoder.

For motif learning, given a graph dataset, a motif-guided subgraph segmenter generates subgraph samples for each graph, and the GNN encoder turns these subgraphs into $d$-dimensional vector representations. We learn graph motifs also as $d$-dimensional vectors through clustering, where K prototypical cluster centers are kept in an embedding table as representations of motifs. Similar and significant subgraphs are assigned to the same motif, and thus moved to be closer to the corresponding motif representation. We train our model in an Expectation-Maximization (EM) fashion to update both the motif assignment of each subgraph and the representations of motifs.

For leveraging learned motifs, we propose a novel graph-to-subgraph contrastive learning framework for GNN pre-training. One of the key components for a successful contrastive learning is to generate semantically meaningful and diverse views of each data instance. For example, a continuous span within a sentence (Joshi et al., 2020) or a random crop of a image (Chen et al., 2020). Previous approaches, such as Deep Graph Infomax and InfoGraph leverage node-to-graph views for contrastive prediction, which is not sufficient enough to capture the subgraph information. To address this issue, we propose to learn graph-to-subgraph contrastive. As motifs can represent the key graph property by its nature, we propose to leverage the learned motifs to generate more informative subgraphs, and benefit GNN contrastive learning. For example, alpha helix and beta sheet can come together as a simple $\beta\beta\alpha$ fold to form a zinc finger protein, and have unique properties. By learning such subgraph co-occurrence via contrastive learning, the pre-trained GNN should capture more subtle structural semantics of graph.

After pre-training on ogbg-molhiv dataset using our proposed *MICRO-Graph*, the framework can successfully learn meaningful motifs, including Benzene ring, Nitro and Acetate, etc. Meanwhile, the pre-trained GNN encoder can be used to fine-tune on seven chemical property prediction benchmarks, yielding 2.0% average improvement over non-pretrained GNNs and outperforming other self-supervised pre-training baselines. Extensive ablation studies further show that the learned motifs are indeed significant for the contrastive learning framework.

## 2 Related Work

The goal of self-supervised learning is to train a neural network model to capture significant characteristics of data without human annotations. This paper studies whether we can use such framework to automatically extract graph motifs, one of the significant structural patterns in graph, and leverage the learned motifs to benefit self-supervised learning. In the following, we first review some preliminaries about graph motifs and the challenge for motif mining, and then review approaches of pre-training graph neural networks in a self-supervised manner.

**Graph motifs** are building blocks of complex graphs. They reveal the interconnections of graphs and represent graph characteristics. Mining motifs can benefit different tasks, from exploratory analysis to graph transfer learning (Henderson et al., 2012). Motif discovery has been studied for many years, and various algorithms have been proposed. There are generally two categories for automatically extracting motifs from graph data, either exact counting as in Milo et al. (2002); Kashtan et al. (2004); Schreiber & Schwöbbermeyer (2005); Chen et al. (2006), or sampling and statistical estimation as in Wernicke (2006). However, both approaches can not scale to large graph datasets with continuous and high-dimension features, which is often the case in real-world applications. In this paper, we proposes to turn the discrete motif mining problem into differentiable cluster learning, and adopt a self-supervised GNNs learning framework to automatically extract important graph motifs. Such method can be generalized to large-scale graph datasets.

Another GNN-based recent work related to graph motifs is the GNNExplainer (Ying et al., 2019). The GNNExplainer focuses on the post-process model interpretation. It can identiy graph substructures that are important for graph property prediciton, e.g. motifs. The difference between GNNExplainer and MICRO-Graph is that the former identify motifs at a single graph level, and the later learn motifs across the whole dataset.

**Contrastive learning** is one of the state-of-the-art self-supervised representation learning algorithms that achieves great results for visual representation learning (Chen et al., 2020; He et al., 2019). Contrastive learning force that embeddings of views generated from the same data (e.g. a image crops) instance to be closer to each other, while those from different instances apart. One of the key component in contrastive learning is to generate informative and diverse views from each data instance. In computer vision, researchers adopt various image augmentation techniques, including crop, color distortion, Gaussian blurs, to generate such views. However, when in comes to graph domain, constructing informative view of graph, e.g., subgraph, is a challenging task. In our framework, we utilize the learned motifs, which are significant subgraph patterns, to guide constructing informative views from each graph, and conduct graph-to-subgraph contrastive learning.

**Self-supervised learning for GNNs** also draws many attention recently. For graphs, representations can be at different levels, e.g. node level and (sub)graph level. Veličković et al. (2018); Hu et al. (2020c); Qiu et al. (2020) mainly focus on node-level representation learning in a single large graph, as opposed to the focus of this paper, representations of the whole graphs. Hu et al. (2020b) provides a systematic analysis of pre-training strategies on graphs for both node-level and graph-level representations. However, only the node-level learning is self-supervised, and annotated labels are utilized to perform supervised learning at the graph level. For graph level self-supervised representation learning, Sun et al. (2019) proposed a contrastive framework InfoGraph to maximize the mutual information between the representations of the whole graphs and node representations. The views they utilize are each node in the graph with the whole graph, which capture more subtle subgraph semantics. In Rong et al. (2020), GROVER model and a motif based self-supervised learning task was proposed, where the discrete motifs are first extracted using a professional software, and then these motifs are used as prediction labels for pre-training the model. GROVER is the state-of-the-art model, but as we discussed above, the motif extraction in GROVER requires domain knowledge and can't scale to large datasets with high-dimensional or continuous features. To alleviate these issues, we propose a novel graph-to-subgraph view construction, and the subgraph generation is guided by learned motifs.

## 3 METHODOLOGY

The goal of this paper is to train a GNN encoder that can automatically extract graph motifs, i.e. significant subgraph patterns. Motif discovery on discrete graph structures is a combinatorial problem, and it is hard to generalize to large datasets with continuous features. We thus propose to formalize this problem as a differentiable clustering learning problem and solve it via self-supervised GNN learning. In this section, we formalize the problem and introduce the overall framework of *MICRO-Graph* in Section 3.1, and then describe each module in details in the following sections.

### 3.1 THE OVERALL FRAMEWORK OF *MICRO-Graph*

Given a dataset with M graphs $\mathbb{G} = \{\mathcal{G}_1, ..., \mathcal{G}_M\}$, the differentiable clustering learning problem is meant to learn two things. One is a GNN-based graph encoder $\boldsymbol{E}(\cdot)$ that maps input (sub)graphs into a D-dimensional embedding space. The other is a K-slot embedding table $\{\boldsymbol{m}_1, ..., \boldsymbol{m}_K\}$, where each slot is a D-dimensional motif vector $\boldsymbol{m}$ corresponding to a cluster center of embeddings of frequently occurred subgraphs. To tackle this problem, we introduce the *MICRO-Graph* framework which consists three modules: 1) a *Motif-guided Segmenter* to extract important subgraphs; 2) a *Motif-Learner* to cluster sampled subgraphs and identify motifs; 3) a constrastive learning module for graph-subgraph contrastive learning.

The overall framework is shown in Figure1. We now describe details of each module in the following sections. We first introduce the *Motif Learner* in section 3.2, and then get to the *Motif-guided Segmenter* and the constrastive learning module in section 3.3 and section 3.4 respectively.

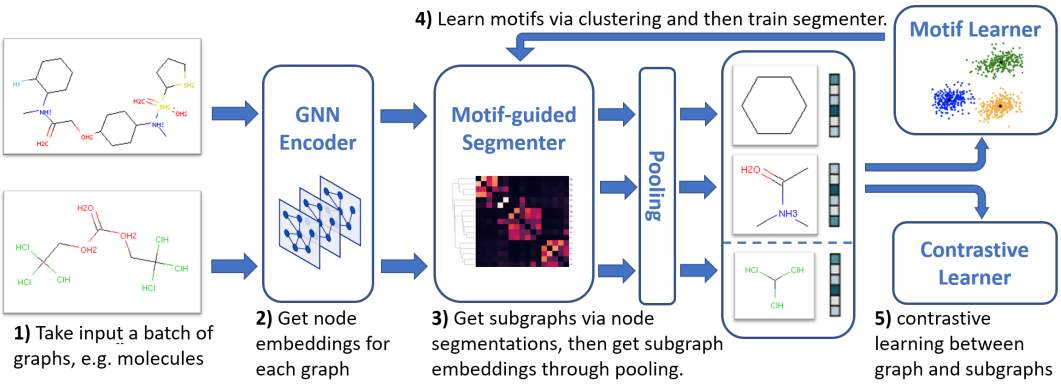


Figure 1: Overall framework of *MICRO-Graph*, which trains a GNN in a self-supervised manner to automatically extract motifs and then leverage the motifs to generate more informative subgraphs for graph-to-subgraph contrastive learning.

### 3.2 MOTIF LEARNER VIA EM CLUSTERING

The *Motif Learner* learns motifs by applying an Expectation-Maximization (EM) style clustering algorithm on sampled subgraphs. The *Motif-guided Subgraph Segmenter* extracts $N$ subgraphs $\{g_j\}_{j=1}^N$ from the input whole graphs. For each subgraph $g_j$, we generate its embedding $\boldsymbol{e}_j = \boldsymbol{E}(g_j)$ and calculate the cosine similarity between $\boldsymbol{e}_j$ and each of the K motif vectors $\{\boldsymbol{m}_1, ..., \boldsymbol{m}_K\}$. We denote the similarity between $\boldsymbol{e}_j$ and the kth motif vector as $S_{k,j} = \phi(\boldsymbol{m}_k)^T \phi(\boldsymbol{e}_j)$[1]. In vector notation, the K-dimensional vector of similarities between $g_j$ and all K motif vectors is denoted as $\boldsymbol{s}_j$, and the K-by-N-dimensional motif-to-subgraph similarity matrix is denoted as $\boldsymbol{S}$, where the $j$-th column of $\boldsymbol{S}$ is $\boldsymbol{s}_j$ and the entry (k, j) of $\boldsymbol{S}$ is $S_{k,j}$.

**E-Step.** The goal of the E-step is to come up with motif-based cluster assignments for subgraph embeddings $\{\boldsymbol{e}_j\}_{j=1}^N$. The assignments can be represented by a K-by-N-dimensional matrix $\boldsymbol{Q} = [\boldsymbol{q}_1, ..., \boldsymbol{q}_N]$, where the $j$-th column $\boldsymbol{q}_j$ contains the probabilities of assigning the $j$-th subgraph to each of the K motifs. When considering hard cluster assignments, each $\boldsymbol{q}_j$ will be a one-hot vector. When considering soft cluster assignments, each $\boldsymbol{q}_j$ will be a probability vector with all the entries sum up to one. This vanilla version clustering problem boils down to maximizing the objective $Tr(\boldsymbol{Q}^T \boldsymbol{S})$, which corresponds to a cluster assignment $\boldsymbol{Q}$ that maximizes similarities between embeddings and its assigned motif. This objective works fine for a traditional EM clustering algorithm when embeddings are fixed. However, as we are doing representation learning, and the embeddings themselves will also change during learning, using this vanilla objective in the E-step can lead to a degenerate solution, i.e. all embeddings collapse to a single cluster center. To avoid this issue, we follow YM. et al. (2020) to introduce an entropy term and an equal-size constraint on $\boldsymbol{Q}$ to make all the clusters to have similar sizes. Our final objective is:

$$\max_{\boldsymbol{Q} \in \mathcal{Q}} Tr(\boldsymbol{Q}^T \boldsymbol{S}) + \frac{1}{\lambda} H(\boldsymbol{Q}) \tag{1}$$

where $H(\boldsymbol{Q}) = -\sum_{i,j} Q_{i,j} \log Q_{i,j}$ is the entropy function, and the constraint set $\mathcal{Q}$ requires the marginal projection of $\boldsymbol{Q}$ onto its columns and rows to be uniform.

$$\mathcal{Q} = \{\boldsymbol{Q} \in \mathbb{R}_+^{K,N} | \boldsymbol{Q}\boldsymbol{1}_{\boldsymbol{N}} = \frac{\boldsymbol{1}_{\boldsymbol{K}}}{K}, \boldsymbol{Q}\boldsymbol{1}_{\boldsymbol{K}} = \frac{\boldsymbol{1}_{\boldsymbol{N}}}{N}\} \tag{2}$$

where $\boldsymbol{1}_{\boldsymbol{N}}$ and $\boldsymbol{1}_{\boldsymbol{K}}$ are all one vectors. This specific constraint optimization problem turns out to be an optimal transportation problem with a closed-form solution as below and can be solved efficiently using a fast Sinkhorn-Knopp algorithm.

$$\boldsymbol{Q}^* = diag(\boldsymbol{u}) \cdot \exp(\lambda \boldsymbol{S}) \cdot diag(\boldsymbol{v}) \tag{3}$$

Here $\boldsymbol{u}$ and $\boldsymbol{v}$ are normalization vectors. The derivations can be found in Cuturi (2013).

[1]For notation simplicity, we denote a L-2 normalization operator $\phi(\cdot)$ such that $\phi(x) = x \,/\, \|x\|_2$.

**M-Step.** The goal of the M-step is to maximize the log-likelihood of our data given the cluster assignment matrix $\boldsymbol{Q}$ estimated in the E-step. We update parameters in the GNN encoder and the motif embedding table through the M-step. This step is equivalent to a supervised K-class classification problem with labels $\boldsymbol{Q}$ and prediction scores $\boldsymbol{S}$. Thus, we first apply a columnwise softmax normalization with temperature $\tau_g$ to $\boldsymbol{S}$ to convert all entries of $\boldsymbol{S}$ to probabilities, i.e. $\tilde{S}_{k,j} = \text{softmax}_k\big(S_{k,j}/\tau_g\big)$. Then we use the negative likelihood as the loss function.

$$\mathcal{L}_m = -\frac{1}{N}\sum_{j=1}^{N}\sum_{k=1}^{K} Q_{k,j}\log \tilde{S}_{k,j} \tag{4}$$

### 3.3 Motif-guided Subgraph Segmenter

Sampling informative subgraphs is crucial for both the *Motif-Learner* and the contrastive learning module . Traditional heuristic approaches such as random walk and k-hop neighbour sampling cannot guarantee to generate semantically reasonable and informative subgraphs. For example, heuristically sampled molecule subgraphs are likely to be a chain of carbons, which doesn't contain much information about the original molecule, or it can be a fragment of a meaningful chemical structure, which loses the original chemical property. Since motifs are by nature informative subgraph patterns, we propose to leverage the learned graph motifs to design the *Motif-guided segmenter*, which will learn to segment a given graph into several subgraphs that are close to some discovered motifs.

**Subgraph Sampling via Segmentation.** To generate subgraphs from a graph $\mathcal{G}_i$ via segmentation, we first generate a node affinity matrix $\boldsymbol{A}^{(i)}$, and then do segmentation based on $\boldsymbol{A}^{(i)}$. Specifically, given the graph $\mathcal{G}_i$ with $n$ nodes, we use the GNN encoder $\boldsymbol{E}(\cdot)$ to generate the D-dimensional node embeddings $\{\boldsymbol{n}_1, ..., \boldsymbol{n}_n\}$ of all n nodes in $\mathcal{G}_i$. After that, we compute the n-by-n-dimensional node affinity matrix $\boldsymbol{A}^{(i)}$ by first computing pairwise cosine similarities between node embeddings, and then applying row-wise softmax normalization with temperature $\tau_n$ to the cosine similarity matrix. This normalization step is important because it transforms all the affinity scores to be in the range (0,1) and make the affinity scores of node pairs with high cosine similarities further stand out. The formula for computing the affinity score between node $s$ and node $t$ in graph $\mathcal{G}_i$, i.e. the entry $(s, t)$ of $\boldsymbol{A}^{(i)}$ is the following.

$$A^{(i)}_{s,t} = \text{softmax}_s\big(\phi(\boldsymbol{n}_s)^T\phi(\boldsymbol{n}_t)/\tau_n\big) \tag{5}$$

Afterwards, we treat this affinity matrix $\boldsymbol{A}^{(i)}$ as a complete graph with n nodes and affinity scores as edge weights. Applying spectral clustering on this complete graph segments nodes into different groups. Within these groups, the connected components that have more than three nodes are collected as our sampled subgraphs. Multiple subgraphs may be sampled from the whole graph $\mathcal{G}_i$. For a particular subgraph $g_j$, its embedding $\boldsymbol{e}_j$ will be generated by indexing and aggregating the node embeddings generated when computing the affinity matrix. For example, if $\mathbb{I}$ is the indices of the nodes forming the subgraph $g_j$ selected by the *Motif-guided segmenter*, then $\boldsymbol{e}_j = Aggregate(\{\boldsymbol{n}_1, ..., \boldsymbol{n}_n\}[\mathbb{I}])$. The aggregate operation can be any order-invariant operation over a set of vectors, e.g. mean, sum, or elementwise max. In our experiment, we follow the start-of-the-art result from previous works and use the mean. Collecting subgraphs from all $M$ whole graphs result in the total set of subgraphs $\{g_j\}_{j=1}^{N}$ mentioned above.

**Motif-guided Training.** To train the segmenter to produce subgraphs close to some motifs, we use the motif-to-subgraph similarity matrix $\boldsymbol{S}$ mention above as the guidance. Consider a subgraph $g_j$ sampled from a whole graph $\mathcal{G}_i$ whose node affinity matrix is $\boldsymbol{A}^{(i)}$. If the similarity between $g_j$ and any motif is higher than a threshold, we would like the affinity value among all the nodes within this subgraph $g_j$ to increase, and the affinity value between these nodes and other nodes not in $g_j$ to decrease. The loss function is as below.

$$\mathcal{L}_s = -\frac{1}{N}\sum_{i=1}^{N}\sum_{(s,t)\in g_j} A^{(i)}_{s,t}\cdot \mathbf{1}\{\exists k \mid \ S_{k,j} > \eta_k\} \tag{6}$$

Here $\eta_k$ is the threshold used to decide whether a subgraph is similar enough to the learned mofit k, and it is dynamically computed. In each iteration, we set $\eta_k$ to select the top 10% most similar subgraphs to motif k.

The intuition is that if the subgraph $g_j$ is considered similar to a motif, then we update the embeddings of its nodes to become similar. By optimizing this loss, during the next sampling round, nodes produced motif-like subgraphs are more likely to be segmented together, which leads to more subgraph samples align with the motifs.

### 3.4 Contrastive Learning between graphs and subgraphs

An expressive GNN encoder $\boldsymbol{E}(\cdot)$ is essential for capturing graph properties and moreover accurately identifying motifs. We thus introduce a constrastive learning module to help guide the GNN learning. This module and the *Motif Learner* will mutually enhance each other to train the GNN encoder to better capture the graph properties.

Contrastive learning is one of the state-of-the-art self-supervised learning methods. One key component in contrastive learning is to generate informative and diverse views of data instances. Previous contrastive methods on graphs utilized either nodes or whole graphs as views, which do not very well capture the micro-structure of graphs. To alleviate this issue in our constrastive learning module, we use the subgraph generated by our *Motif-guided segmenter* as one view of the graph, and the whole graph as another view.

Similarly to how we generate the subgraph embeddings, the whole graph embedding $\boldsymbol{h}_i$ of a graph $\mathcal{G}_i$ is generated by aggregating node embeddings output by the GNN encoder $\boldsymbol{E}(\cdot)$, $\boldsymbol{h}_i = Aggregate(\{\boldsymbol{n}_1, ..., \boldsymbol{n}_n\})$. Then we construct the M-by-N dimensional graph-to-subgraph similarity matrix $\boldsymbol{W}$, which is cosine similarity between each graph-subgraph pair followed by a row-wise softmax normalization with temperature $\tau_g$.

$$W_{i,j} = \text{softmax}_i\big(\phi(\boldsymbol{h}_i)^T \phi(\boldsymbol{e}_j)/\tau_g\big) \tag{7}$$

For the whole graph $\mathcal{G}_i$, subgraphs sampled from it are considered as positive pairs to it, and subgraphs sampled from other graphs are considered as negative pairs to it. Then the contrastive objective function is the following:

$$\mathcal{L}_c = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{N} W_{i,j} \cdot \mathbf{1}\{g_j \in \mathcal{G}_i\} \tag{8}$$

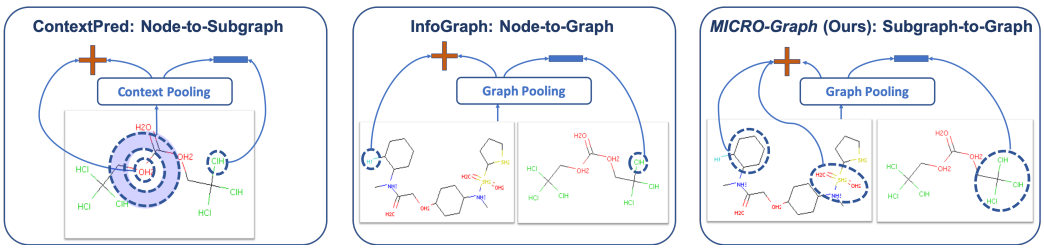


Figure 2: Different ways to generate informative and diverse views for graph-data. Context Prediction adopts nodes and context subgraphs as views, and it hopes each node has similar embedding with its context subgraph while dissimilar to other node's context subgraph; InfoGraph uses nodes and whole graphs views, and thus learns via node-to-graph contrastive. Our method considers subgraphs as high-level views of each graph to conduct graph-to-subgraph contrastive learning, and thus the GNN can capture higher-order and global structure information.

We show the main difference for generating views of the original graph between our framework and other existing methods in Figure 2. Most existing contrastive learning methods are conducted on node-node views or node-graph views. Researchers in computer vision domain have found that conducting contrastive learning on more flexible views, e.g. an arbitrary-sized image crop, leads to better representations, especially much better than views at the pixel level (Chen et al., 2020). In

our framework, the contrastive learning on graph-subgraph views is similar to this intuition from computer vision. It can capture higher-level structural patterns the node views cannot capture, and thus produce more meaningful representations.

Note that the subgraphs we utilize is generated by our *Motif-guided Segmenter*. In section 4, we systematically study the influence of subgraph sampling methods. The experiment results show that simple heuristic sampling approaches lead to really bad generalization performance. This shows the significance of leveraging the learned graph motifs to generate more informative views of graph-structured data.

### 3.5 JOINT TRAINING

The overall objective of *MICRO-Graph* is a weighted sum of all three loss terms described above.

$$\mathcal{L} = \lambda_m \mathcal{L}_m + \lambda_s \mathcal{L}_s + \lambda_c \mathcal{L}_c \tag{9}$$

With the proper constraints introduced to the E-step of the *Motif-Learner* (equation (1) and (2)), the whole framework can be trained end-to-end without worrying about degenerate solutions.

We show the the pseudocode of our approach in Algorithm 1. We first initialize the motif vectors and GNN encoder (line 2 - 3). Then for each batch of graphs $\mathbb{G}$, our segmenter will calculate the node-node affinity matrices $\boldsymbol{A}$, and extract the subgraphs based on the affinity scores (line 5 - 6). Then we apply GNN message passing on whole graphs to get the node embeddings, and aggregate them for both graph and subgraph embeddings (line 7 - 9). The next step is to compute the motif-to-subgraph similarity matrix $\boldsymbol{S}$, based on that we can compute both the threshold $\eta$ and cluster assignment matrix $\boldsymbol{Q}$ (line 11 - 13). With all these values, our three loss terms the final joint loss can be computed accordingly (line 15 - 20).

**Algorithm 1** Pseudocode of MICRO-Graph in PyTorch Style, full version in Appendix A

```
1  # temperature parameters: tau_g, tau_n, weight parameters: lamb_m, lamb_c, lamb_s
2  model = Motif(args) # model contains all motif vectors, model.motifs
3  encoder = GNN(args) # GNN encoder for pre-training
4  for G in loader:
5      #I: node index of subgraph,  A: affinity matrices, num_subs: # of subgraphs per graph
6      I, num_subs, A = segmenter(G, encoder)
7      n = encoder(G)
8      h = aggregate(n, G) # M x D
9      e = aggregate(n, G, I) # N x D
10     S = pairwise_cosine_sim(model.motifs, e) # K x N
11     with torch.no_grad():
12         eta = topk_threshold(S) # take the topk similarity scores
13         Q = sinkhorn(S) # use the Sinkhorn-Knopp to solve for Q
14
15     loss_s = sampler_loss(A, eta)
16     S_tilde = torch.softmax(S / tau_g, dim=0) # K x N
17     loss_m = motif_loss(Q, S_tilde)
18     W = torch.softmax(pairwise_cosine_sim(h, e)/tau_g, dim=1)
19     loss_c = contrastive_loss(W, num_subs) # need num_subs from segmenter
20     loss = lamb_m*loss_m + lamb_c*loss_c + lamb_s*loss_s
```

*MICRO-Graph* can simultaneously train an expressive GNN encoder and learn the underlying motifs of the given graph dataset. Moreover, the motif learning and contrastive learning are mutually reinforced: a better GNN encoder can help get more accurate subgraph embeddings and thus help motif mining, while a better motif can help generate more informative graph-to-subgraph views and help contrastive learning.

## 4 EXPERIMENTS

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

### 4.1 EXPERIMENT SETUP

We mainly focus on chemical property prediction task. Specifically, we pre-train GNNs using *MICRO-Graph* on the ogbg-molhiv dataset from Open Graph Benchmark (OGB) (Hu et al., 2020a). This dataset contains 40K chemical compounds with average graph size 25. We test our pretrained model on other smaller molecule-related graph classification datasets from OGB. For more details of the datasets, please see Appendix E.

#### 4.1.1 EVALUATION PROTOCOLS

After we pre-train a GNN encoder on ogbg-molhiv dataset, we evaluate its effectiveness using the following two evaluation protocols:

**Transfer Learning Setting**: we fine-tune the pre-trained GNN model with a small portion of labels of the downstream task. We adopt the same train-test and model selection procedure as in Yanardag & Vishwanathan (2015); Zhang et al. (2018); Xu et al. (2018), where we perform 10-fold cross-validation and select the epoch with the best cross-validation accuracy averaged over the 10 folds. We follow the OGB official setting to use the ROC-AUC as our metric.

**Feature Extraction Setting**: the setting is almost the same as transfer learning. Except that we fix the pre-trained GNN, use it as feature extractor to get graph representations of all the data in downstream tasks, and then train a linear classifiers on top.

#### 4.1.2 BASELINES

For transfer learning setting, we adopt non-pretrain and direct supervised learning as the baseline. We also use three state-of-the-art GNN Self-Supervised Learning (SSL) methods as baselines:

**InfoGraph** (Sun et al., 2019) maximizes the mutual information between the representations of the whole graphs and the representations of its substructures at different granularity.

**Context prediction** (Hu et al., 2020b) uses subgraphs to predict their surrounding graph structures, so that nodes appearing in similar structural contexts will be mapped to nearby representations.

**GPT-GNN** (Hu et al., 2020c) predicts masked edges and masked node attributes. The edge prediction makes node representations to be close when there are edges between them. The attribute prediction captures how node attributes are distributed over all graphs.

**GROVER** (Rong et al., 2020) first uses professional software, e.g. RDKit(Landrum et al., 2006), to extract functional groups (motifs) from a whole dataset. Using these motifs as a label set, each molecule is assigned a label representing which motif shows up in it and which doesn't. The model is then pre-trained by predicting this motif label as a multi-class classificaiton problem.

#### 4.1.3 MODEL CONFIGURATION AND HYPERPARAMETERS

We use the Deeper Graph Convolutional Networks (DeeperGCNs) proposed in Li et al. (2020), the state-of-the-art GNN architectures, as our base GNN encoder. For fair comparison, we use the same model hyperparameters (number of layers is 5 and number of hidden dimension is 300) recommended in Li et al. (2020) for our method and all baselines. We pre-train our model using Adam optimizer for 100 epochs, with batch size (number of graphs per batch) as 512. During fine-tuning, we train the model for 100 epochs with batch size as 32, and select the output model with highest validation performance.

### 4.2 EVALUATION RESULT

The evaluation results under transfer learning setting and feature extraction setting is illustrated in Table 1 and Table 2. For both setting, the average performance of our proposed *MICRO-Graph* is higher than all the self-supervised learning baselines, and it achieves the highest results on most datasets. For transfer learning setting, we can gain about 2.0% performance enhancement against non-pretrain baseline. This shows the effectiveness of our self-supervised learning framework for pre-training GNNs, especially when the label data is scarce.

### 4.3 ABLATION STUDY

In the following, we conduct a series of ablation studies to systematically analyze how the motif learning can benefit the contrastive learning.

| 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) |
| MICRO-Graph | **76.16 ± 2.51** | **83.78 ± 1.77** | 77.50 ± 3.35 | **75.51 ± 0.67** | **57.28 ± 1.09** | 76.68 ± 0.36 | **65.42 ± 0.62** | **73.19 (+2.0)** |

Table 1: Transfer learning performance (ROC-AUC) of MICRO-Graph compared with other self-supervised learning (SSL) baselines on molecule property prediction benchmarks. Pre-train GNNs on ogbg-molhiv dataset, fine-tune the pre-trained model on each downstream task for 10 times.

| 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 | 54.7 ± 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 | 68.2 ± 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 |
| MICRO-Graph | **69.54 ± 0.39** | **81.07 ± 0.42** | 63.69 ± 0.56 | **72.74 ± 0.15** | **55.39 ± 0.26** | **72.91 ± 0.12** | **61.04 ± 0.07** | **68.05** |

Table 2: Feature extraction performance (ROC-AUC) of MICRO-Graph compared with other self-supervised learning (SSL) baselines on molecule property prediction benchmarks. Use pre-trained models to extract graph representations for each data and train linear classifiers on top. Run each experiment 5 times.

#### 4.3.1 WHETHER MOTIF IS HELPFUL FOR SUBGRAPH SAMPLING?

As previously discussed, the main difference of our contrastive framework with existing works is that we leverage the subgraph-to-graph view as contrastive prediction task. As there are multiple ways to generate subgraphs, we thus study whether our proposed motif-guided subgraph segmenter can indeed help contrastive learning.

We implement three widely adopted heuristic subgraph sampling baseline: random walk (RW) and K-hop neighbours sampling (K-hop). We replace our Motif-guided Subgraph Segmenter (MSS) component with these two heuristic sampling algorithm, and conduct experiments under the Transfer Learning setting. All the other settings are the same.

As is shown in Table 3, replacing our Motif-guided Subgraph Segmenter (MSS) with heurstic subgraph sampler will significantly influence the performance. With random walk or k-hop sampling, subgraph-to-graph contrastive learning can only bring in 0.3-0.5% average performance enhancement against non-pretrain, which is far less than the one with MSS. This shows that the key of our overall performance enhancement of is not only based on utilizing the subgraph-to-graph view, but also highly depends on the informative subgraphs generated with the help of motifs. For the details of sample strategies and the subgraphs sampled by each strategies, please refer to Appendix D.

#### 4.3.2 WHETHER MOTIF NUMBER WILL INFLUENCE CONTRASTIVE LEARNING?

Number of motifs is an important hyperparameter in our motif mining framework. We thus conduct ablation study with three different number of motifs, 5, 20, and 100. As is illustrated in Table 4,

| Sampler | bace | bbbp | clintox | hiv | sider | tox21 | toxcast | Average |
|---|---|---|---|---|---|---|---|---|
| RW | 73.61 ± 2.53 | 82.24 ± 1.99 | 75.63 ± 2.86 | 73.06 ± 1.29 | 55.88 ± 1.69 | 76.14 ± 0.56 | 63.44 ± 0.76 | 71.42 (+0.23) |
| K-hop | 73.24 ± 2.65 | 82.65 ± 1.78 | 76.76 ± 3.88 | 73.48 ± 1.41 | 55.67 ± 1.51 | 76.01 ± 0.69 | 63.34 ± 0.94 | 71.59 (+0.4) |
| MSS | **76.16 ± 2.51** | **83.78 ± 1.77** | **77.50 ± 3.35** | **75.51 ± 0.67** | **57.28 ± 1.09** | **76.68 ± 0.36** | **65.42 ± 0.62** | **73.19 (+2.0)** |

Table 3: Ablation study: analyzing the influence of subgraph sampler.

| # Motif | bace | bbbp | clintox | hiv | sider | tox21 | toxcast | Average |
|---|---|---|---|---|---|---|---|---|
| 5 | $75.81 \pm 2.38$ | $82.65 \pm 1.97$ | $76.95 \pm 2.44$ | $74.53 \pm 1.12$ | $56.78 \pm 1.64$ | $\mathbf{77.01 \pm 0.89}$ | $64.45 \pm 0.59$ | 72.59 (+1.4) |
| 20 | $\mathbf{76.16 \pm 2.51}$ | $\mathbf{83.78 \pm 1.77}$ | $\mathbf{77.50 \pm 3.35}$ | $\mathbf{75.51 \pm 0.67}$ | $57.28 \pm 1.09$ | $76.68 \pm 0.36$ | $\mathbf{65.42 \pm 0.62}$ | **73.19 (+2.0)** |
| 100 | $75.98 \pm 2.36$ | $83.68 \pm 1.62$ | $76.91 \pm 2.57$ | $75.06 \pm 0.92$ | $\mathbf{57.32 \pm 1.12}$ | $76.98 \pm 0.55$ | $65.39 \pm 0.69$ | 73.04 (+1.85) |

Table 4: Ablation study: analyzing the influence of different motif numbers.

with different number of motifs, our MICRO-Graph framework can consistently enhance the transfer performance by a large margin. Among the three numbers used, the middle one (20) gives the best result on average.

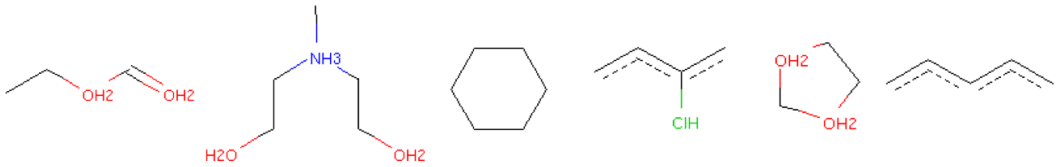


Figure 3: Top-6 frequently occurred motifs, represented by their closest subgraph.

### 4.4 Visualization of the Learned Motifs

We further shows some learned motifs by collecting the closest subgraph to them. As is illustrated in Figure 3, the *MICRO-Graph* automatically learns some motifs that are similar to important function groups in molecule domain, such as Benzene ring and Acetate. This shows that our framework can indeed learn reasonable and meaningful motifs. For a complete list of all the learned motifs, please refer to Appendix C.

## 5 Conclusion

In this paper, we propose *MICRO-Graph* to pre-train GNN in a self-supervised manner that can automatically extract graph motifs from large-scale graph datasets. In addition, the learned motifs can guide to generate more informative subgraphs, and help conduct graph-to-subgraph contrastive learning. The motif learning and contrastive learning are mutually reinforced, and eventually help pre-train a more generalizable GNN encoder. By pre-training on ogbn-molhiv molecule dataset with our proposed *MICRO-Graph*, we can learn meaningful motifs that fit existing molecular functional groups. Meanwhile, the pre-trained GNN can be used to fine-tune on seven chemical property prediction benchmarks, yielding 2.8% average improvement over non-pretrained GNNs and outperforming other self-supervised pre-training baselines.

## References

Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive graph-level representation learning via graph-graph proximity, 2019.

Jin Chen, Wynne Hsu, Mong Li Lee, and See-Kiong Ng. Nemofinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 106–115, 2006.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances, 2013.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1231–1239, 2012.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020a.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=HJlWWJSFDH.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020c.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77, 2020. URL https://transacl.org/ojs/index.php/tacl/article/view/1853.

Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Greg Landrum et al. Rdkit: Open-source cheminformatics. 2006.

Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns, 2020.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Nicolò Navarin, Dinh V. Tran, and Alessandro Sperduti. Pre-training graph neural networks with kernels, 2018.

Carl O Pabo, Ezra Peisach, and Robert A Grant. Design and selection of novel cys2his2 zinc finger proteins. *Annual review of biochemistry*, 70(1):313–340, 2001.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Graph contrastive coding for graph neural network pre-training. In *KDD*, 2020.

Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data, 2020.

Falk Schreiber and Henning Schwöbbermeyer. Frequency concepts and pattern detection for the analysis of motifs in networks. In *Transactions on computational systems biology III*, pp. 89–104. Springer, 2005.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization, 2019.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax, 2018.

Lichen Wang, Bo Zong, Qianqian Ma, Wei Cheng, Jingchao Ni, Wenchao Yu, Yanchi Liu, Dongjin Song, Haifeng Chen, and Yun Fu. Inductive and unsupervised representation learning on graph structured objects. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkem91rtDB`.

Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM transactions on computational biology and bioinformatics*, 3(4):347–359, 2006.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374, 2015.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks, 2019.

Asano YM., Rupprecht C., and Vedaldi A. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

# A PSEUDOCODE OF THE MAIN ALGORITHM

```
# temperature parameters: tau_g, tau_n
# weight parameters: lamb_m, lamb_c, lamb_s

model = Motif(args) # model contains all motif vectors, model.motifs
encoder = GNN(args) # GNN encoder for pretraining

for G in loader:
    # sample subgraphs from each whole graph and return the node
    # indices of these subgraphs => I
    # return how many subgraphs have been sampled from each
    # whole graph => num_subs
    # compute the node-node affinity matrix A,
    # return the sum of affinity of nodes within each subgraph => sum_A
    # I: list of len N, num_subs: M x 1, sum_A: N x 1
    I, num_subs, sum_A = segmenter(G, encoder)

    # encode the input graph to get node embeddings
    n = encoder(G)

    # pool all node embeddings together for the whole graph embedding
    h = aggregate(n, G) # M x D

    # pool nodes belong to the subgraph for the subgraph embedding
    e = aggregate(n, G, I) # N x D

    # compute motif-subgraph similarity
    S = pairwise_cosine_sim(model.motifs, e) # K x N

    with torch.no_grad():
        # compute similarity threshold eta as the top 10% for each motif
        S_top10, _ = S.topk(k=int(0.1 * S.shape[1]), dim=1) # K x 0.1*N
        eta = S_top10[:, -1] # K x 1
        Q = sinkhorn(S) # use the Sinkhorn-Knopp algorithm to solve for Q, K x N

    # identy whether a subgraph is similar enough to a motif
    S_mask = (S > eta).sum(dim=0) > 0 # 1 x N

    # compute the sampler loss
    loss_s = sum_A[S_mask]

    # normalize S
    S_tilde = torch.softmax(S / tau_g, dim=0) # K x N

    # compute motif learning loss
    loss_m = - (Q * S_tilde.log()).sum(dim=0).mean()

    # compute pairwise similarities between each
    # whole graph embeddings and subgraph embeddings
    W = torch.softmax(pairwise_cosine_sim(h, e)/ tau_g, dim=1)

    # compute the contrastive loss
    blocks = [torch.ones(1, int(n)) for n in num_subs]
    W_mask = torch.block_diag(*blocks)
    loss_c = - (W_mask * W.log()).sum(dim=1).mean()

    # final loss
    loss = lamb_m*loss_m + lamb_c*loss_c + lamb_s*loss_s


def segmenter(G, encoder):
    with torch.no_grad():
        I = []
        num_subs = []
        sum_A = []
        for G_i in G:
            # encode the input graph to get node embeddings
            n = encoder(G_i)

            # compute the node-node affinity matrix A_i
            A_i = torch.softmax(pairwise_cosine_sim(n, n) / tau_n, dim=1)

            # apply spectral clustering and find connected components
            # to segment subgraphs, I_i is a list
            I_i = find_connected_components(spectral_clustering(A_i))

            # the node indices of these subgraphs
            I += I_i

```

```
            # how many subgraphs are sampled from each whole graph
            num_subs += [len(I_i)]

            # the sum of affinity values of nodes within each subgraph
            sum_A += [(A_i[index][:, index]).sum() for index in I_i]

        return I, num_subs, sum_A


def sinkhorn(S, num_iters=3, lamb=20):
    '''
    Implementation of the sinkhorn function adopted from
    https://github.com/facebookresearch/swav/blob/master/main_swav.py
    '''
    with torch.no_grad():
        Q = torch.exp(S).t()
        Q /= torch.sum(Q)
        u = torch.zeros(Q.shape[0]).to(Q.device)
        r = torch.ones(Q.shape[0]).to(Q.device) / Q.shape[0]
        c = torch.ones(Q.shape[1]).to(Q.device) / Q.shape[1]

        curr_sum = torch.sum(Q, dim=1)
        for it in range(num_iters):
            u = curr_sum
            Q *= (r / u).unsqueeze(1)
            Q *= (c / torch.sum(Q, dim=0)).unsqueeze(0)
            curr_sum = torch.sum(Q, dim=1)
        return (Q / torch.sum(Q, dim=0, keepdim=True)).t().float()
```

## B NOTATION SUMMARY

Below, we summary the important notations and symbols used paper in the order they appeared.

| | |
|---|---|
| Graph level | |
| $M$ | Total number of whole graphs |
| $\mathcal{G}_i$ | Whole graphs |
| $\boldsymbol{h}_i$ | Whole graph embeddings |
| Subgraph level | |
| $N$ | Total number of subgraphs |
| $g_j$ | Subgraphs |
| $\boldsymbol{e}_j$ | Subgraph embeddings |
| Node level | |
| $\boldsymbol{n}_i$ | Node embedding |
| $\boldsymbol{A}^{(i)}$ | Node affinity matrix, n-by-n dimensional for a whole graph with n nodes |
| $\boldsymbol{A}^{(i)}_{s,t}$ | Node affinity between node s and node t in a graph |
| $\mathbb{I}$ | Indices of nodes forming a subgraph selected by the *Motif-guided Segmenter* |
| Motifs | |
| $K$ | Number of motifs |
| $\boldsymbol{m}, \boldsymbol{m}_k$ | Motif vectors |
| $\boldsymbol{S}$ | Similarities between all K motifs and all N subgraphs, K-by-N dimensional |
| $\boldsymbol{s}_j$ | Similarities between all K motifs and the subgraph j, K-by-1 dimensional |
| $S_{k,j}$ | Similarity between the motif k and the subgraph j, scalar |
| $\tilde{S}_{k,j}$ | Normalized similarity between the motif k and the subgraph j |
| $\boldsymbol{Q}$ | Motif-based cluster assignment matrix, K-by-N dimensional |
| $\boldsymbol{q}_j$ | Motif-based cluster assignment of subgraph j, 1-by-N dimensional |
| $\eta_k$ | Threshold for deciding whether a subgraph is similar enough to motif k |
| Contrastive | |
| $\boldsymbol{W}$ | Normalized similarities between M graphs and N subgraphs, M-by-N dimensional |
| $W_{i,j}$ | Normalized similarities between graph i and subgraph j |
| Others | |
| $\tau_n$ | Temperature for the softmax normalization of node-node similarity |
| $\tau_g$ | Temperature for the softmax of motif-subgraph and graph-subgraph similarity |
| $\boldsymbol{E}(\cdot)$ | GNN encoder used to generate node embeddings and (sub)graph embeddings |
| $D$ | Dimension of node embeddings, (sub)graph embeddings, and motif vectors |

## C TOP K CLOSESET SUBGRAPHS TO LEARNED MOTIFS

Examples of the first 10 learned motifs of the ogbg-molhiv dataset is shown in Figure 4 and 5.

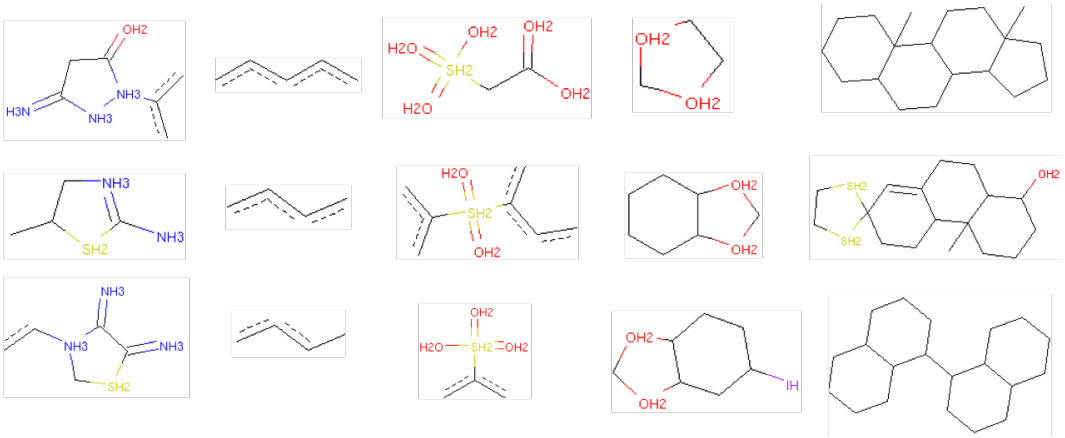


Figure 4: Motif 1-5, represented by top k closest subgraphs to the learned motif representations. Each row represents a motif, represented by some subgraphs that is closest to these motifs. Three columns indicates top 1, top 2, and top 3 most similar subgraph respectively.

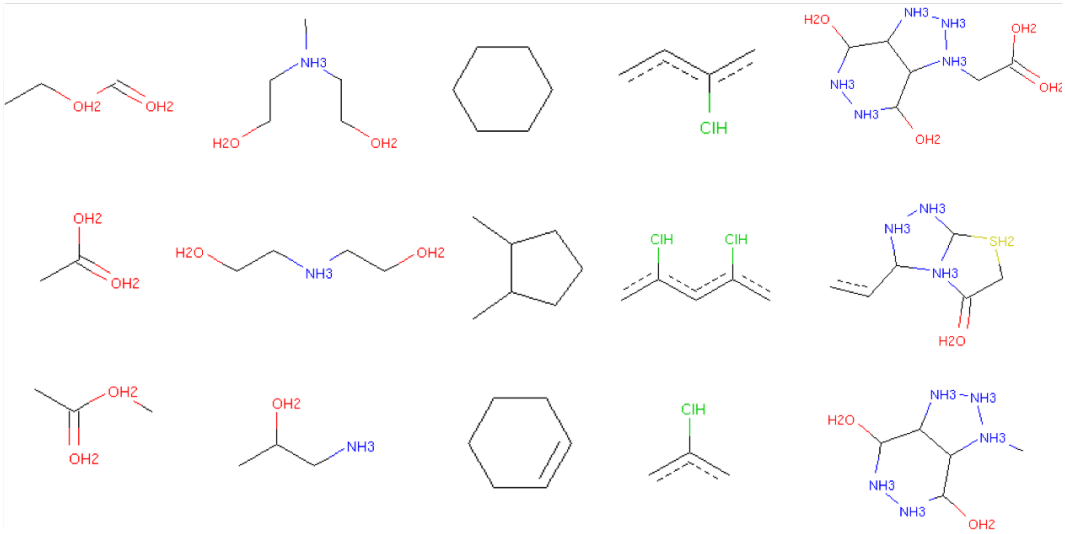


Figure 5: Motif 6-10, represented by top k closest subgraphs to the learned motif representations. Each row represents a motif, represented by some subgraphs that is closest to these motifs. Three columns indicates top 1, top 2, and top 3 most similar subgraph respectively.

## D SAMPLING STRATEGIES AND SAMPLED SUBGRAPHS

Here we describe the details of our heuristic sampling strategies.

For random walk, we use a random walk length uniform in [10, 40]. Starting from a randomly selected seed node, we randomly select its neighborhood as next hop, until reaching the walk length threshold.

For $K$-hop neighbors, we pick hop number $k$ to be 1 or 2 with equal probability. Starting randomly selected seed node, we collect all the neighbors within $k$ hop as the sampled subgraph.

We also shows some subgraph examples generated by these two heuristic strategies and our proposed motif-guided subgraph segmenter in Figure 6. From the sampled subgraphs, we can see that random walk is more likely to generate chains, while k-hop sampling is more likely to generate half part of a Benzene ring. Neither of these two heuristic approaches can successfully generate a complete and clean functional group, and the generated subgraphs are not that meaningful. On the contrary, our motif-guided sampler can succssfully generate a complete benzene ring and two other molecule substructures. This intuitively explains why the graph-to-subgraph contrastive learning can only work with our proposed subgraph segmenter.

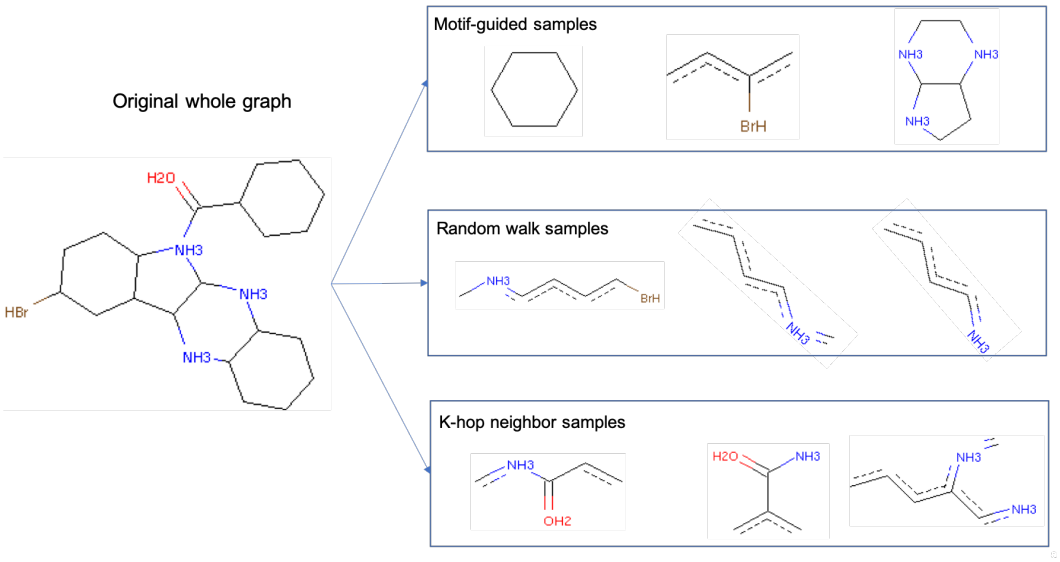


Figure 6: Comparison between different sampling strategies. The original graph is shown on the left. Samples produced by three different sampling strategies are shown on the right. The top row shows the samples by our motif-guided segmenter. The following two rows corresponding to random walk samples and k-hop samples.

## E CHEMICAL PROPERTY PREDICTION BENCHMARKS

In our experiments, we evaluated model performance on seven Open Graph Benchmark (OGB) molecule property prediction datasets. We provide a synopsis of each downstream task dataset from Hu et al. (2020b) below:

- **bace:** Qualitative binding results for a set of inhibitors of human $\beta$-secretase 1.
- **bbbp:** Blood-brain barrier penetration (membrane permeability).
- **clintox:** Qualitative data classifying drugs approved by the FDA and those that have failed clinical trials for toxicity reasons.
- **hiv:** Experimentally measured abilities to inhibit HIV replication.
- **sider:** Database of marketed drugs and adverse drug reactions (ADR), grouped into 27 system organ classes.

| Dataset | bace | bbbp | clintox | hiv | sider | tox21 | toxcast |
|---|---|---|---|---|---|---|---|
| # graphs | 1513 | 2039 | 1477 | 41127 | 1427 | 7831 | 8576 |
| # nodes | 51577 | 49068 | 38637 | 1049163 | 48006 | 145459 | 161088 |
| # edges | 111536 | 105842 | 82372 | 2259376 | 100912 | 302190 | 161088 |
| # tasks | 1 | 1 | 2 | 1 | 27 | 12 | 617 |

Table 5: Statistics on number of graphs, nodes, edges, and tasks in each OGB molecule dataset.

- **tox21:** Toxicity data on 12 biological targets, including nuclear receptors and stress response pathways.
- **toxcast:** Toxicology measurements based on over 600 in vitro high-throughput screenings.

Table 5 summarizes important statistics of the OGB molecule datasets related to the number of graphs, the size of graphs, and number of properties that require prediction for each molecule. For further information on the OGB datasets (e.g. node and edge features), please refer to Hu et al. (2020b) and Hu et al. (2020a).

## F CONTEXT PREDICTION BASELINE HYPERPARAMETERS

We show the hyper-parameters we used for running the context prediction baseline in Table 6. We show additional experiments of ContextPred with different parameters in Table 7.

| | |
|---|---|
| Context represent mode | cbow |
| Context pooling | mean |
| Negative sample ratio | 1 |
| Context size | 3 |
| r1 | 4 |
| r2 | 7 |

Table 6: Hyper-parameters of the context prediction pretraining

| | bace | bbbp | clintox | hiv | sider | tox21 | toxcast | Average |
|---|---|---|---|---|---|---|---|---|
| r1=2, r2=5 | $73.55 \pm 2.5$ | $81.7 \pm 2.84$ | $75.82 \pm 4.11$ | $73.79 \pm 0.9$ | $54.98 \pm 1.48$ | $75.6 \pm 0.78$ | $63.88 \pm 0.76$ | 71.33 |
| r1=3, r2=4 | $72.65 \pm 2.32$ | $81.23 \pm 1.98$ | $73.14 \pm 6.8$ | $73.67 \pm 0.99$ | $53.99 \pm 1.38$ | $75.67 \pm 0.65$ | $63.53 \pm 0.79$ | 70.55 |

Table 7: Additional experiments for ContextPred with different parameters r. Note: results shown in Section 4 are with r1 = 4 and r2 = 7

## G LINEAR EVALUATION IN ACCURACY

We show the linear evaluation result in **accuracy** of our model and the baselines in Table **??**. Note that the results of tables in Section 4 are in AUC-ROC rather than accuracy.

| SSL methods | bace | bbbp | clintox | hiv | sider | tox21 | toxcast | Average |
|---|---|---|---|---|---|---|---|---|
| ContextPred | $57.13 \pm 0.55$ | $80.58 \pm 0.21$ | $\mathbf{93.03 \pm 0.00}$ | $\mathbf{96.49 \pm 0.00}$ | $75.35 \pm 0.08$ | $92.21 \pm 0.01$ | $83.68 \pm 0.02$ | 82.64 |
| InfoGraph | $67.51 \pm 1.83$ | $82.08 \pm 1.13$ | $90.92 \pm 0.40$ | $93.24 \pm 1.92$ | $68.80 \pm 0.52$ | $89.60 \pm 0.37$ | $80.19 \pm 0.06$ | 81.76 |
| GPT-GNN | $59.33 \pm 0.26$ | $73.82 \pm 2.06$ | $93.01 \pm 0.07$ | $94.16 \pm 3.06$ | $70.86 \pm 0.34$ | $88.60 \pm 0.26$ | $81.20 \pm 0.12$ | 80.14 |
| MICRO-Graph | $\mathbf{76.49 \pm 0.25}$ | $\mathbf{85.44 \pm 0.20}$ | $93.01 \pm 0.05$ | $94.49 \pm 0.00$ | $\mathbf{75.82 \pm 0.00}$ | $\mathbf{92.74 \pm 0.00}$ | $\mathbf{84.24 \pm 0.05}$ | **86.32** |

Table 8: Feature extraction performance (ACC) of MICRO-Graph compared with other self-supervised learning (SSL) baselines on molecule property prediction benchmarks. Use pretrainedmodels to extract graph representations for each data and train linear classifiers on top. Run eachexperiment 5 times

## H A SYNTHETIC DATASET TO STUDY MOTIF LEARNING

In addition to studying the pre-training in chemical domain, we also construct a synthetic dataset to that align with our assumptions to verify the effectiveness of the propose method. We assume there exist $K$ graph motifs, and each whole graph can be representated by certain combinations of these motifs. Following this rule, we first select some graph structures as motifs, and randomly sample some combinations, and generate graphs (as is illustrated in Figure 7) by combining the motifs, with some randomly added or deleted nodes and edges. Each graph will also be assigned a corresponding one-hot vector label, whose dimension is the total number of combinations we generated for the dataset. Eight subgraph templates and examples of generated whole graphs are shown below. As is illustrated in Figure 8, our *MICRO-Graph* can successfully learn the underlying graph motifs and templates without any annotations.

The advantage of constructing such synthetic dataset is that we can know the underlying ground-truth of graph motifs and combination rules. We believe on top of this toy dataset, more complex syntax and semantic knowledge can be incorporated for automatic motif and even rule mining communities.

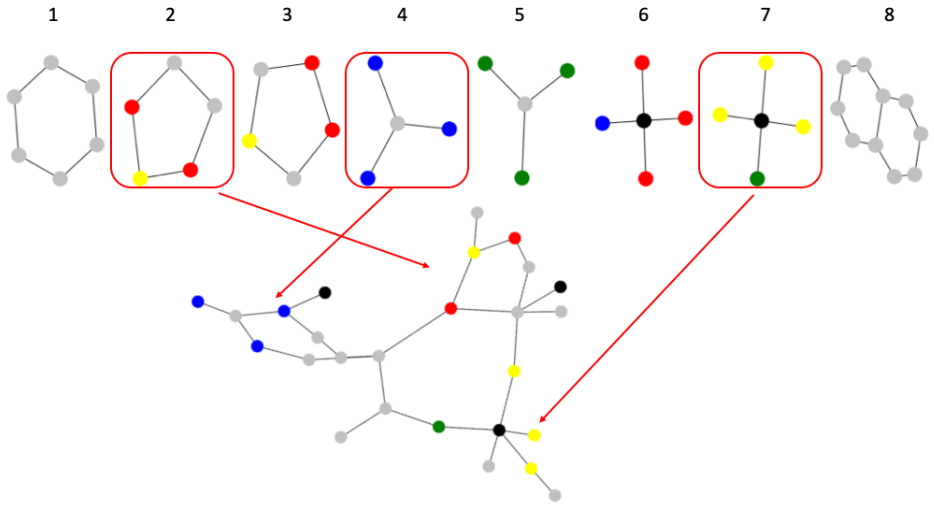


Figure 7: Example of a synthetic graph. The upper 8 graphs are the base motifs, and we generate this graph with a combination [2, 4, 7]. Different colors represent different node features.

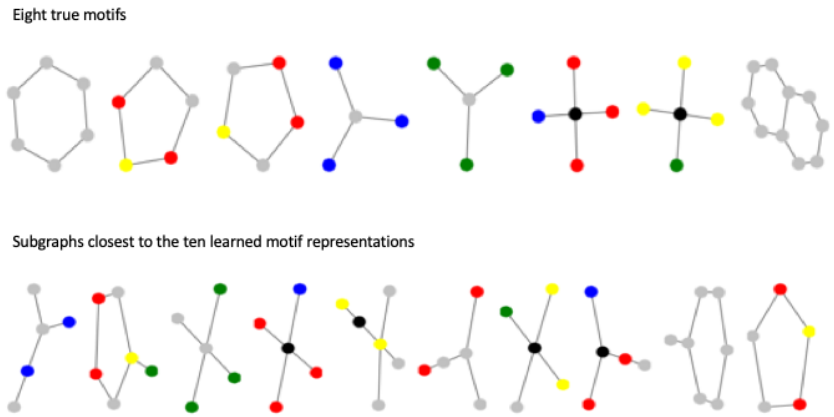


Figure 8: Comparison of the eight motifs used to generate the synthetic dataset and subgraphs closest to the ten learned motif representations