A Limitations

Our results and analysis on the graph tokenizer and graph decoder are confined to the task of MGM pretraining. Different tokenizers and decoders might offer advantages for other generative modeling methods, such as autoregressive modeling [32].

SGTs [39, 40] are limited in expressive power for graph structures compared to standard GNNs, like GINs [37]. Theoretically, the separation of expressiveness power between SGTs and standard GNNs grows exponentially in the GNN's depth [40]. However, SGTs exhibit comparable, if not better, performances to pretrained GNN-based tokenizers, as demonstrated in Table 3b. We attribute this intriguing observation to two key factors. Firstly, SGTs (*i.e.,* simple GNNs) are still powerful and can "distinguish almost all non-isomorphic graphs" [40]. They have shown decent results in practice [39, 60]. Secondly, we conjecture that a better pretraining method for GNN-based tokenizers could exist, but current pretraining techniques do not fully harness the potential of GNNs in their roles as effective tokenizers. Indeed, the significant difference in performance between GraphCL and VQ-VAE (Table 3b) emphasizes the impact of pretraining methods on the tokenizer's performance. We leave the investigation of how to effectively pretrain GNN-based tokenizers as future works.

B Related Works

We have included the literature review of MGM in the main body of the paper. Here we elaborate on the literature review in the following areas.

Molecule SSL with motifs. Motifs are statistically significant subgraph patterns [32, 61], and have been applied in existing molecule SSL methods. Autoregressive pretraining methods [32, 1, 61] generate motifs instead of nodes at each generation step, in order to improve the validity of the generated molecules. Motifs are also used in contrastive learning [17, 62, 18]. Sun *et al.* [17] substitute motifs within a molecule with their chemically similar counterparts to create high-quality augmentations. [62, 18] construct molecules' views at the motif-level to supplement the original views at the atom-level. In predictive pretraining, Rong *et al.* [2] pretrain a graph encoder to predict the FGs inside the molecule. These previous works have developed extensive molecule fragmentation methods for motif discovery. However, these fragmentation methods have been overlooked as tokenizers in MGM pretraining. Our work addresses this gap by summarizing the common fragmentation rules and examining the performances of the selected fragmentation methods in MGM pretraining.

Data tokenization. Tokenization is a data pre-processing technique that divides the original data into smaller elements and converts them to tokens. It is widely used in NLP to split sentences into word-level units [13, 63, 64]. Due to the surging interests in Transformers [65], tokenization is also applied on images [15, 16] and audios [66, 67]. Tokenization fragments these data into sequences of patches to fit the shapes of transformer's input and output. A tokenizer can be designed by heuristics [68], incorporating domain knowledge [69], and pretraining on the target dataset [70, 19, 42]. In this work, we study graph tokenizers, which are less explored in previous works.

Relations to contrastive learning. When using a GNN-based tokenizer, MGM involves minimizing the distances between the outputs from two network branches (*i.e.,* the tokenizer branch and the autoencoder branch). At first glance, this design might seem similar to the contrastive learning methods of BYOL [71], SimSiam [72], and BGRL [47], which also minimize the output differences between two network branches. However, a closer inspection reveals several critical distinctions between MGM and these methods. Firstly, MGM feeds uncorrupted data to the tokenizer branch and feeds corrupted data to the autoencoder branch, encouraging the autoencoder to reconstruct the missing information. In contrast, BYOL, SimSiam, and BGRL use corrupted data in both of their branches, constituting different training objectives. Secondly, while BYOL, SimSiam, and BGRL employ nearly identical architectures for their two branches, MGM can adopt distinctly different architectures for its autoencoder and tokenizer. In our best-performing experiment, the autoencoder has more than ten layers of GNNs and Transformers, while the tokenizer is a shallow single-layer network (Table 3). Finally, MGM employs remask decoding to constrain the encoder's ability on reconstruction, which is not used in contrastive learning methods [71, 72, 47].

Subgraph-enhanced Graph Neural Network. Subgraph-enchanced GNN [73, 74, 75] refers to an emerging class of GNNs that fragments a graph into subgraphs before encoding, in order to improve the GNNs' expressivenss [76, 77, 78]. The common graph fragmentation method is node-wise, such

that each fragmented subgraph is associated with a unique node in the original graph. For example, ESAN [73] obtains subgraphs by sampling ego-networks or deleting one node from the original graph. Given the subgraphs, subgraph-enchanced GNNs generate node embeddings in every subgraph by applying a series of equivariant message-passing layers [73, 74]. Finally, these embeddings are pooled to output the graph embedding. Our work is related to subgraph-enhanced GNNs that we also study graph fragmentation. The major distinction is that we focus on using the tokens derived from these fragmented graphs as the reconstruction targets in MGM for molecules.

C Pseudo Code

We present the pseudocode of SimSGT. This code uses a single-layer SGT of GIN as an example.

Algorithm 1 Pytorch style pseudocode of SimSGT

```
## phi: graph encoder
## rho: graph decoder

def SGT(g, embed):
 ## SGT: a single-layer GIN tokenizer
 x, edge_index = g

 # message passing
 x = propagate(embed(x), edge_index) + (1+eps) * embed(x)

 # batch normalization layer
 x = batchnorm(x)
 return x


for g in loader:
 # random masking
 g_hat, m_pos = random_masking(g) # m_pos: mask positions

 # tokenization. embed is a linear layer
 y = SGT(g, phi.embed).detach() # detach: stop-gradient

 # autoencoder forward
 y_hat = rho(remask(phi(g_hat), m_pos))

 # minimize loss
 loss = distance_loss(y_hat[m_pos], y[m_pos])
 loss.backward()
```

D Experimental Setup

Computational resource. We perform experiments on an NVIDIA DGX A100 server. Each individual experiment can be run on a single GPU without exceeding 30 GBs of GPU memory.

D.1 Compared Methods

Motif-based tokenizers. We now elaborate on the details of the two compared motif-based tokenizers:

- **MGSSL** [1] employs the BRICS [35] method for molecule fragmentation (Section 2.2). To obtain more fine-grained fragments, MGSSL employs two additional rules to break the BRICS's output fragments: 1) separate the single atoms attached to cycles; 2) if a connected subgraph comprising three or more atoms is not part of a cycle, break it down as a new fragment.
- **RelMole** [18] combines the fragmentation functions of Cycles and FGs for molecule fragmentation (Section 2.2). Further, it extracts the carbon-carbon single bonds that are not covered in the previous step as new fragments.

We use the motif vocabulary provided by their paper for molecule fragmentation. Given a molecule, we convert its fragmented motifs into one-hot encodings, which serve as the reconstruction targets.

Pretrained GNN-based tokenizers. We use the atomic numbers as the node features and exclude edge features in pretrained GNN-based tokenizers. We show in Appendix E that incorporating edge

Table 9: Experimental settings for pretraining on 2 million molecules from ZINC15 and fine-tuning on eight datasets in MoleculeNet: BBBP, Tox21, ToxCast, Sider, ClinTox, MUV, HIV, and Bace.

(a) Node and edge features.

Type		Range
Node features	atomic numbers	1∼118
	chirality tag	{unspecified, tetrahedral cw, tetrahedral ccw, other}
Edge features	bond type	{single, double, triple, aromatic}
	bond direction	{-, endupright, enddownright}

(b) Hyperparameters.

Encoder	pretrain			fine-tuning		
	lr	batch size	epochs	lr	batch size	epochs
GINE	1e-3	1024	100	{1e-3, 1e-4}	32	100
GTS	1e-4	2048	100	{1e-4, 1e-5}	32	100

features in GNN tokenizers can decrease performance. Due to the removal of edge features, the tokenizer uses the architecture of GIN [37] instead of GINE [12]. We have reported performances of GNN-based tokenizers that are pretrained by GraphCL [4], GraphMAE [9], and VQ-VAE [10, 42]. The implementation of VQ-VAE follows [10] and groups the latent codes by atomic numbers. We strictly follow the procedure in the mentioned papers to pretrain GNNs, which are later used as tokenizers.

Simple GNN-based tokenizers (SGTs). An SGT uses the node feature of atomic number. It uses the graph encoder's linear embedding function of atomic numbers. We present the graph operators for our tested SGTs below:

$$\text{GIN:} \quad \omega(\mathbf{A}) = \mathbf{A} + (1+\epsilon)\mathbf{I}, \tag{17}$$

$$\text{GCN:} \quad \omega(\mathbf{A}) = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}, \tag{18}$$

$$\text{GraphSAGE:} \quad \omega(\mathbf{A}) = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}, \tag{19}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$; ϵ is set to 0.5 empirically.

Baselines. We now describe the details of our reported baseline methods:

- **Infomax** [44] learns node representations by maximizing the mutual information between the local summaries of node patches and the patches' graph-level global summaries.
- **ContextPred** [12] uses the embeddings of subgraphs to predict their context graph structures.
- **InfoGraph** [79] conducts graph representation learning by maximizing the mutual information between graph-level representations and local substructures of various scales.
- **GraphCL** [4] performs graph-level contrastive learning with combinations of four graph augmentations, namely node dropping, edge perturbation, subgraph cropping, and feature masking.
- **JOAO** [45] proposes a framework to automatically search proper data augmentations for GCL.
- **AD-GCL** [46] applies adversarial learning for adaptive graph augmentation to remove the redundant information in graph samples.
- **GraphLOG** [48] leverages clustering to construct hierarchical prototypes of graph samples. They further contrast each local instance with its corresponding higher prototype for contrastive learning.
- **RGCL** [49] trains a rationale generator to identify the causal subgraph in graph augmentation. Each graph's causal subgraph and its complement are leveraged in contrastive learning.
- **BGRL** [47] trains an online encoder by learning to predict the output of a target encoder. The target encoder shares the same architecture as the online encoder and is updated through exponentially moving average. The inputs of the online encoder and the target encoder are two different graph augmentations.

Table 10: Experimental setting for pretraining on the 50 thousand molecules from the GEOM dataset and fine-tuning on the four molecular property prediction (regression) datasets and DTA datasets.

(a) Node and edge features.

Type		Range
Node features	atomic numbers	1∼ 118
	chirality tag	{unspecified, tetrahedralcw, tetrahedralccw, other}
	node degree	0∼10
	formal_charge	-5∼5
	number of H	0∼8
	number of radical e	0∼4
	hybridization	{sp, sp2, sp3, sp3d, sp3d2}
	is aromatic	{false, true}
	is in ring	{false, true}
Edge features	bond type	{single, double, triple, aromatic}
	bond stereo	{stereonone, stereoz, stereoe, stereocis, stereotrans, stereoany}
	is conjugated	{false, true}

(b) Hyperparameters and their search spaces. We use the performance on the validation set for hyperparameter tuning. **Bold** indicates the final value used in experiments.

Encoder	pretrain			fine-tune (regression)			fine-tune (DTA)		
	lr	batch size	epochs	lr	batch size	epochs	lr	batch size	epochs
GINE	1e-3	1024	100	1e-3	{32, 128, **256**}	100	{1e-4, **2e-4**}	128	500
GTS	1e-4	1024	300	{1e-4, 2e-4, **3e-4**}	32	100	{1e-4, **2e-4**}	128	500

- **GraphMAE** [9] shows that a linear classifier is insufficient for decoding node types. It applies a GNN for decoding and proposes remask to decouple the functions of the encoder and decoder in the autoencoder.
- **GraphMVP** [52] uses a contrastive loss and a generative loss to connect the 2-dimensional view and 3-dimensional view of the same molecule, in order to inject the 3-dimensional knowledge into the 2-dimensional graph encoder.
- **S2GAE** [50] randomly masks a portion of edges of graphs and pretrain the graph encoder to predict the missing edges.
- **GraphMAE2** [51] applies multi-view random re-mask decoding as a regularization for MGM pretraining.
- **Mole-BERT** [10] combines a contrastive learning objective and a masked atom modeling objective for MRL. Specifically, they observe that mask atom prediction is an overly easy pretraining task. Therefore, they employ a GNN tokenizer pretrained by VQ-VAE [42] to generate more complex reconstruction targets for masked atom modeling.

D.2 Experimenets in Section 4 and Table 5

Here we elaborate the experimental setting for pretraining on 2 million molecules from ZINC15 [43] and fine-tuning on the eight classification datasets in MoleculeNet [28]: BBBP, Tox21, ToxCast, Sider, ClinTox, MUV, HIV, and Bace. This setting covers the experiments in Section 4 and Table 5.

Molecule representations. For SimSGT and other compared methods, we follow previous works [12, 4] and use a minimal set of molecule features as the graph representations (Table 9a). These features unambiguously describe the two-dimensional structure of molecules.

Hyper-parameters. Table 9b summarizes the hyper-parameters. We use different hyper-parameters given different graph encoders. The architectures of the two graph encoders are borrowed from previous works: GINE [12] and GTS [27]. We use large batch sizes of 1024 and 2048 to speed up pretraining. We do not use dropout during pretraining. During fine-tuning, we 50% dropout in GINE layers and 30% dropout in transformer layers. The learning rate for the MUV dataset is 10 times smaller than other datasets. Following [4, 48], we report the last epoch's test performance. We

Table 11: Hyperparameters for fine-tuning on the QM datasets.

QM Dataset	batch size	lr
QM7	32	4e-4
QM8	256	1e-3
QM9	256	1e-3

report the mean performances and the standard deviations across 10 random seeds. Baselines are reproduced using the same setting.

Linear probing experiments. Here we elaborate on the settings of our linear probing experiments (Figure 5b and Figure 7). Specifically, we randomly split the 2 million molecules from ZINC15 into train set (90%) and test set (10%). We train the MGM models on the train set and save the encoder's checkpoint every epoch. The linear classifiers are trained for 1000 epochs on the encoder's frozen hidden representations. We train linear classifiers using 25600 molecule samples from the training set and evaluate them on the whole test set.

- **Probing masked atom types (Figure 5b).** We let linear classifiers predict the masked atom types using the masked atoms' hidden representations. During linear probing, we disable remask-v2 to obtain the masked atoms' hidden representations. Molecules are randomly masked by 0.35 during probing. We use accuracy (%) as the evaluation metric.
- **Probing FGs (Figure 7).** Following [2], we extract 85 types of FGs for each molecule using RDkit [30]. FGs are represented by 85-dimensional binary vectors, whose each dimension indicates the presence of a certain FG. Afterward, we train multi-label linear classifiers on the frozen encoder's mean pooling outputs for FG prediction. Molecules are not masked during probing. We use ROC-AUC (%) as the evaluation metric.

D.3 Experiments in Table 6

We present the experimental setting for pretraining on the 50 thousand molecules from GEOM [54] and fine-tuning on the four molecule property prediction (regression) datasets and two DTA datasets. Our experimental setting follows that in [52]. This setting covers the experiments in Table 6.

Molecule representations. In the graph autoencoder, we use 9-dimensional node features and 3-dimensional edge features of molecules provided by the OGB [80] package, following GraphMVP [52]. The features are summarized in Table 10a. Note that, our tokenizer uses only the atomic numbers as node features and does not use edge features.

Hyper-parameters. The hyperparameters are summarized in Table 10b. We tune the hyperparameters in the fine-tuning stage using the validation performance. Following [52], we report the test performance at the epoch selected by the validation performance. We do not use dropout during pretraining. During fine-tuning, we 50% dropout in GINE layers and 30% dropout in transformer layers.

For a fair comparison, we reproduce Mole-BERT [10]'s performance by pretraining on the 50 thousand molecule samples from the GEOM dataset [54]. The original Mole-BERT is trained on a larger dataset of 2 million molecules from ZINC15 [43].

D.4 Experiments in Table 7

The hyperparameters for fine-tuning on the QM datasets are reported in Table 11.

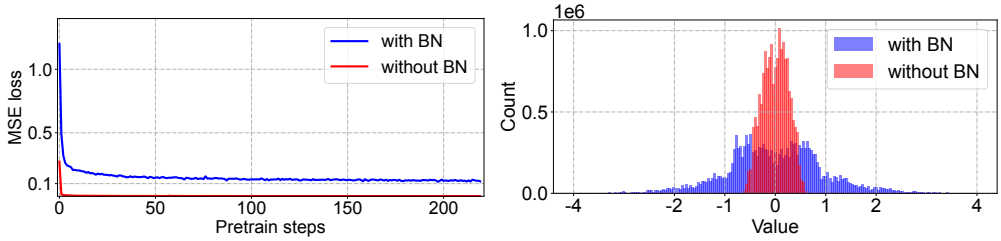
E More Experimental Results

In this section, we provide more experimental results. If not especially noted, these experiments employ an autoencoder of GTS encoder and GTS-Small decoder with remask-v2, and a tokenizer of a single-layer SGT of GIN. Other settings follow that in Appendix D.2.

Influence of edge features for pretrained GNN-based tokenizer. We ablate the impact of the "bond type" and "bond direction" edge features in pretrained GNN-based tokenizers. We use GINE and

Table 12: Average transfer learning ROC-AUC (%) scores on the eight classification datasets in MoleculeNet. Including edge features in tokenizers decreases the performance.

Tokenizer	Edge feature	Tokenizer GNN's depth 1	2	3	4	5
Pretrain, GraphCL	✗	**75.1**	74.5	**74.2**	**74.0**	**74.6**
	✓	74.2	**74.7**	73.7	73.8	73.2
Pretrain, GraphMAE	✗	**75.1**	**74.9**	**74.9**	**75.4**	**75.2**
	✓	74.6	74.6	74.3	74.6	75.0



(a) MSE loss curve with respect to pretrain steps. (b) Histograms for the values of the tokenizer's output.

Figure 8: SimSGT pretraining on the 2 millions molecules from ZINC15.

GIN for the tokenizer with and without edge features. Table 12 shows that including edge features in GNN-based tokenizers negatively influences the transfer learning performance. Therefore, we exclude edge features from pretrained GNN-based tokenizers in our experiments.

Influence of Batch Normalization layers in SGTs. The Batch Normalization [41] (BN) layers in SGTs are crucial to avoid loss vanishment. Figure 8a presents a comparison between SGT "with *vs* without BN". Without the BN layer, the MSE loss drops to lower than 0.01 within a few steps of pretraining. Such small loss values lead to significant model underfitting.

As shown by Figure 8b, the token values of SGT without BN follow a sharp distribution: the values are primarily distributed around zero, and their standard deviation (std) is smaller than 0.35. This minor std issue might be caused by the smoothing effect of GNNs [81]. An expressive neural network (*i.e.,* a graph autoencoder) can quickly fit this sharp target distribution and minimize the loss to a negligible value, causing the problem of loss vanishment. However, if a BN layer is used, it forces each dimension of the tokenizer output to have an std of 1.00, so as to "spread out" the distribution of the SGT tokens. These new SGT tokens of a larger std are harder to fit. They keep the MSE loss at a reasonable range of $0.10 \sim 0.15$ (Figure 8a).

Mask ratio. We apply random node masking throughout the experiments [12]. Figure 9 presents SimSGT's sensitivity with respect to the mask ratios. SimSGT is not sensitive to mask ratios such that a wide range of ratios ($0.25 \sim 0.45$) can generate competitive performances. The ratio of 0.35 achieves the best performance. This ratio is much lower than that for images, where a ratio of 0.75 can generate promising performances [15].

Balancing the distribution of reconstruction targets. As shown in Figure 11, the popularly used ZINC15 dataset includes 12 types of atoms, and 95% of the atoms are distributed on the top three atom types. This skewed distribution renders the node-level token reconstruction an easy pretraining task [10]. Figure 10 shows that the accuracy of predicting node-level tokens converges quickly. Such an easy pretraining task can lead to suboptimal performance as suggested by existing SSL literature [82, 83]. In Figure 11, we show that the induced subgraphs of a single-layer SGT (*i.e.,* one-hop rooted subtrees) follow a more balanced distribution than the distribution of nodes. SGT tokens also have a larger vocabulary size: ZINC15 includes 555 types of one-hop rooted subtrees. Consequently, the accuracy of predicting tokens of a single-layer SGT takes more epochs to converge (Figure 10).

Pooling Method for Subgraph Representations. In previous experiments, we use mean pooling to obtain the subgraph representations for motif-based tokenizers, following the method of obtaining graph representations in [4, 12]. Here we add results for MGSSL tokenizer using sum and max

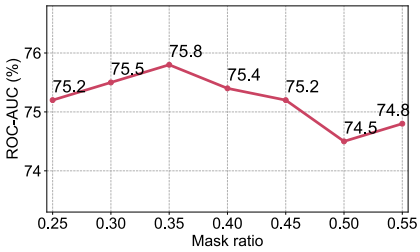


Figure 9: Average ROC-AUC (%) scores with respect to different node masking ratios. The performances are evaluated on the eight classification datasets in MoleculeNet.

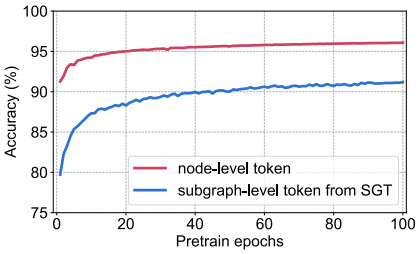


Figure 10: Token prediction accuracies. SGT token prediction is conducted by calculating the Euclidean distance between the autoencoder's output and the vocabulary of all SGT tokens.

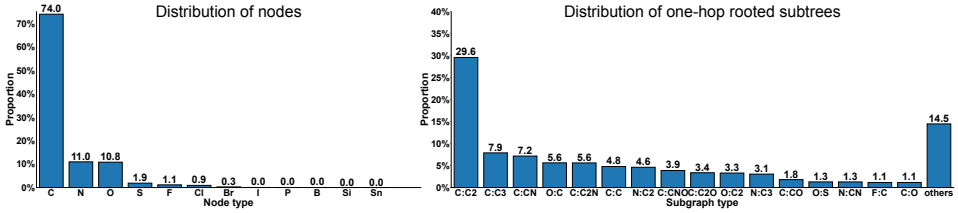


Figure 11: Distributions of graph fragments in MGM. Statistics come from molecules in ZINC15 [43]. For the subgraph types, a colon ':' separates the center node and the neighbor nodes. For example, C:CN denotes a subgraph with a center of a Carbon and neighbors of a Carbon and a Nitrogen.

pooling in Table 13. The results show that mean pooling yields the highest performance, affirming the soundness of our previous experiments.

Full results of Section 4. We provide the full results of experiments in Section 4. Table 14 contains the full results of Table 3a and Table 4. Table 15 contains the full results of Table 3b and Figure 6.

References

[1] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. Motif-based graph self-supervised learning for molecular property prediction. In *NeurIPS*, pages 15870–15882, 2021.

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

[3] Zhiyuan Liu, Sihang Li, Yanchen Luo, Hao Fei, Yixin Cao, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. Molca: Molecular graph-language modeling with cross-modal projector and uni-modal adapter. In *EMNLP*, 2023.

[4] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. In *NeurIPS*, 2020.

[5] Laurianne David, Amol Thakkar, Rocío Mercado, and Ola Engkvist. Molecular representations in ai-driven drug discovery: a review and practical guide. *Journal of Cheminformatics*, 12(1):1–22, 2020.

[6] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272. PMLR, 2017.

[7] Chaochao Yan, Qianggang Ding, Peilin Zhao, Shuangjia Zheng, Jinyu Yang, Yang Yu, and Junzhou Huang. Retroxpert: Decompose retrosynthesis prediction like A chemist. In *NeurIPS*, 2020.

Table 13: ROC-AUC (%) scores on eight MoleculeNet datasets. Compared methods use GTS encoder and GTS-Small decoder with remask-v2 decoding.

Dataset	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	Avg.	GAIN
Motif, MGSSL, Mean	72.5±0.9	77.5±0.4	65.2±0.6	60.7±0.9	85.0±3.5	79.9±1.5	78.0±1.5	83.0±1.0	75.2	5.3
Motif, MGSSL, Max	71.5±0.9	75.8±1.2	66.2±0.7	60.7±1.3	82.6±2.3	78.9±1.8	76.5±1.4	83.8±1.6	74.5	4.6
Motif, MGSSL, Sum	71.7±1.4	75.9±0.6	66.1±0.7	60.4±1.4	83.4±1.5	79.5±1.0	76.8±1.2	84.2±1.1	74.8	4.8

Table 14: Transfer learning ROC-AUC (%) scores on the eight classification datasets in MoleculeNet.

Encoder	Decoder	Remask	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	Avg.
GINE	Linear	-	72.5±0.8	76.0±0.4	63.7±0.5	60.1±0.7	81.2±2.6	74.2±1.6	78.0±0.8	79.6±1.4	73.2
GINE	GINE-Small	-	70.9±0.6	75.1±0.5	63.5±0.4	61.0±0.4	79.1±2.6	76.0±0.5	76.3±0.5	82.5±0.9	73.0
GINE	GINE-Small	v1	70.2±1.0	76.4±0.6	64.2±0.4	61.9±0.8	80.9±1.8	77.8±1.1	78.1±1.1	83.6±1.1	**74.1**
GTS	Linear	-	72.9±1.0	76.6±0.8	63.8±0.8	58.3±1.3	81.9±5.5	78.5±1.5	78.0±2.0	83.1±0.9	74.1
GTS	GTS-Small	-	72.0±0.6	74.7±0.4	63.7±0.4	58.9±0.6	86.0±2.0	78.9±1.2	77.3±0.7	81.0±0.7	74.1
GTS	GTS-Small	v1	71.3±1.0	77.0±1.2	66.2±0.6	60.6±1.4	84.5±3.4	81.5±1.5	77.0±1.6	83.5±1.2	75.2
GTS	GTS-Small	v2	72.2±0.9	76.8±0.9	65.9±0.8	61.7±0.8	85.7±1.8	81.4±1.4	78.0±1.9	84.3±0.6	**75.8**
GTS	GTS-Tiny	v2	71.9±1.2	77.2±1.1	65.6±0.5	61.7±1.4	82.9±2.4	79.6±1.4	76.8±1.3	82.1±1.5	74.7
GTS	GTS	v2	70.7±1.2	76.4±0.9	66.1±0.4	60.3±0.9	84.7±4.6	79.6±0.7	76.8±1.9	84.5±0.8	74.9

[8] Umit V Ucak, Islambek Ashyrmamatov, Junsu Ko, and Juyong Lee. Retrosynthetic reaction pathway prediction through neural machine translation of atomic environments. *Nature communications*, 13(1):1186, 2022.

[9] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *KDD*, pages 594–604. ACM, 2022.

[10] Jun Xia, Chengshuai Zhao, Bozhen Hu, Zhangyang Gao, Cheng Tan, Yue Liu, Siyuan Li, and Stan Z. Li. Mole-BERT: Rethinking pre-training graph neural networks for molecules. In *The Eleventh International Conference on Learning Representations*, 2023.

[11] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 10871–10880. PMLR, 2020.

[12] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics, 2019.

[14] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pages 15979–15988. IEEE, 2022.

[16] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: BERT pre-training of image transformers. In *ICLR*. OpenReview.net, 2022.

[17] Mengying Sun, Jing Xing, Huijun Wang, Bin Chen, and Jiayu Zhou. Mocl: Data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In *KDD*, pages 3585–3594. ACM, 2021.

[18] Zewei Ji, Runhan Shi, Jiarui Lu, Fang Li, and Yang Yang. Relmole: Molecular representation learning based on two-level graph similarities. *Journal Chemical Information Modeling*, 62(22):5361–5372, 2022.

Table 15: Transfer learning ROC-AUC (%) scores on the eight classification datasets in MoleculeNet.

Tokenizer	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	Avg.
Node	70.3±0.9	76.4±1.0	65.7±0.7	61.7±0.9	81.9±3.5	79.8±0.7	77.4±1.8	84.6±1.1	74.7
Motif, MGSSL	72.5±0.9	77.5±0.4	65.2±0.6	60.7±0.9	85.0±3.5	79.9±1.5	78.0±1.5	83.0±1.0	75.2
Motif, RelMole	71.4±1.3	77.1±0.4	66.3±0.6	58.9±1.2	80.7±2.7	79.2±1.4	78.0±1.0	83.6±1.0	74.4
Pretrain, GraphCL, 1 layer GIN	72.2±1.4	76.8±0.4	66.0±0.8	60.8±1.2	81.4±2.5	81.1±1.5	78.4±1.3	83.8±0.9	75.1
Pretrain, GraphCL, 2 layer GIN	70.8±0.6	76.7±0.8	66.3±0.4	60.6±1.2	84.3±3.3	77.2±1.4	76.3±2.1	83.8±1.2	74.5
Pretrain, GraphCL, 3 layer GIN	70.6±0.8	77.1±0.6	65.4±0.5	59.3±1.4	81.2±3.8	79.4±2.3	76.4±1.9	84.2±1.1	74.2
Pretrain, GraphCL, 4 layer GIN	72.1±0.9	76.9±0.6	65.7±0.7	59.6±0.9	77.6±3.5	81.7±1.2	77.6±2.2	81.1±1.1	74.0
Pretrain, GraphCL, 5 layer GIN	71.4±0.7	76.9±0.7	66.5±0.7	60.0±1.2	80.8±2.3	81.0±1.1	77.7±1.2	82.5±1.5	74.6
Pretrain, VQ-VAE, 1 layer GIN	72.2±0.9	77.0±0.6	66.5±0.6	61.3±1.8	82.8±3.7	79.1±2.0	77.4±1.5	84.2±0.8	75.1
Pretrain, VQ-VAE, 2 layer GIN	71.5±0.8	76.6±0.6	65.9±0.7	60.3±0.7	82.1±2.0	81.5±2.4	77.2±1.9	84.3±1.1	74.9
Pretrain, VQ-VAE, 3 layer GIN	71.9±0.9	76.7±0.9	65.8±0.8	61.2±1.8	79.5±2.5	80.0±0.9	78.0±1.3	81.7±0.9	74.4
Pretrain, VQ-VAE, 4 layer GIN	72.5±1.0	77.0±0.6	66.3±0.3	61.7±1.5	86.7±2.2	80.3±1.6	77.6±1.3	82.7±0.9	75.6
Pretrain, VQ-VAE, 5 layer GIN	72.0±0.9	76.8±0.6	65.6±0.6	61.5±1.1	84.3±1.3	80.6±1.0	78.1±1.4	81.9±0.8	75.1
Pretrain, GraphMAE, 1 layer GIN	72.0±1.1	77.3±0.6	66.3±0.3	60.9±1.5	83.7±2.7	79.9±1.4	76.3±2.2	84.0±1.6	75.1
Pretrain, GraphMAE, 2 layer GIN	71.9±0.8	77.6±0.6	66.0±0.5	60.8±1.8	81.9±3.9	79.0±1.6	76.5±2.4	85.3±0.7	74.9
Pretrain, GraphMAE, 3 layer GIN	71.4±0.7	77.6±0.8	65.8±0.3	61.1±1.7	82.2±2.9	79.2±1.5	77.4±2.1	84.3±0.9	74.9
Pretrain, GraphMAE, 4 layer GIN	72.3±0.7	76.6±0.7	66.1±0.8	62.0±1.2	83.3±2.1	80.1±2.2	77.9±1.7	85.0±0.9	75.4
Pretrain, GraphMAE, 5 layer GIN	72.6±0.6	76.4±0.5	65.7±0.6	62.4±1.3	84.0±2.8	80.0±1.3	78.7±1.3	81.5±1.3	75.2
SGT, 1 layer GIN	72.2±0.9	76.8±0.9	65.9±0.8	61.7±0.8	85.7±1.8	81.4±1.4	78.0±1.9	84.3±0.6	75.8
SGT, 2 layer GIN	71.3±0.7	77.0±0.9	66.2±0.8	61.5±0.8	84.9±2.0	80.7±2.0	78.1±1.1	84.5±0.8	75.5
SGT, 3 layer GIN	71.7±0.8	77.6±0.8	66.2±0.6	61.2±2.4	85.8±2.6	80.4±1.1	77.7±2.1	84.1±1.4	75.6
SGT, 4 layer GIN	72.0±0.9	77.1±1.2	65.4±1.0	61.7±1.5	83.8±2.2	80.1±1.9	77.5±1.2	84.7±0.9	75.3
SGT, 5 layer GIN	70.7±0.7	77.0±0.7	65.9±0.8	61.1±1.6	83.1±1.6	79.9±1.5	77.6±1.6	83.9±1.4	74.9
SGT, 1 layer GCN	71.9±0.9	77.8±1.0	66.5±0.7	62.0±0.8	85.2±1.6	79.2±1.4	77.9±2.3	84.4±2.1	75.6
SGT, 3 layer GCN	71.1±1.0	77.4±0.8	66.3±0.4	61.6±1.1	84.4±2.5	78.7±1.1	77.7±2.4	85.1±1.3	75.3
SGT, 5 layer GCN	71.0±1.0	76.5±0.6	65.6±0.3	61.9±1.3	83.7±2.1	78.3±2.0	76.7±1.2	84.9±1.0	74.8
SGT, 1 layer GraphSAGE	72.5±0.7	77.1±0.8	66.7±0.5	61.3±1.0	86.2±1.8	80.7±1.5	76.6±1.7	83.5±1.0	75.6
SGT, 3 layer GraphSAGE	70.3±1.1	77.9±0.7	65.4±0.6	60.3±1.0	87.4±3.7	79.2±1.9	77.6±2.1	84.9±0.5	75.4
SGT, 5 layer GraphSAGE	71.6±1.0	76.5±0.7	66.4±0.7	60.8±1.4	85.6±2.4	78.6±1.3	77.0±1.6	84.4±0.9	75.1

[19] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, pages 8821–8831, 2021.

[20] Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, and George Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1036–1050, 2005.

[21] Jonathan Clayden, Nick Greeves, and Stuart Warren. *Organic chemistry*. Oxford university press, 2012.

[22] Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin Zheng, and Weiqiang Wang. Maskgae: Masked graph modeling meets graph autoencoders. *CoRR*, abs/2205.10053, 2022.

[23] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. MGAE: marginalized graph autoencoder for graph clustering. In *CIKM*, pages 889–898. ACM, 2017.

[24] Sixiao Zhang, Hongxu Chen, Haoran Yang, Xiangguo Sun, Philip S. Yu, and Guandong Xu. Graph masked autoencoders with transformers. *CoRR*, abs/2202.08391, 2022.

[25] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. Adaptive graph encoder for attributed graph embedding. In *KDD*, pages 976–985. ACM, 2020.

[26] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. Mask and reason: Pre-training knowledge graph transformers for complex logical queries. In *KDD*, pages 1120–1130. ACM, 2022.

[27] Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. In *NeurIPS*, pages 13266–13279, 2021.

[28] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[29] Tapio Pahikkala, Antti Airola, Sami Pietilä, Sushil Shakyawar, Agnieszka Szwajda, Jing Tang, and Tero Aittokallio. Toward more realistic drug-target interaction predictions. *Briefings Bioinform.*, 16(2):325–337, 2015.

[30] Greg Landrum. Rdkit documentation. *Release*, 1(1-79):4, 2013.

[31] Elena S Salmina, Norbert Haider, and Igor V Tetko. Extended functional groups (efg): an efficient set for chemical characterization and structure-activity relationship studies of chemical compounds. *Molecules*, 21(1):1, 2015.

[32] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2328–2337. PMLR, 2018.

[33] Daylight Chemical Information Systems, Inc. *Daylight Theory Manual*. http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html.

[34] Jonathan Clayden, Nick Greeves, and Stuart Warren. *Organic chemistry*. Oxford university press, 2012.

[35] Jörg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem: Chemistry Enabling Drug Discovery*, 3(10):1503–1507, 2008.

[36] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[37] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[38] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022.

[39] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 2019.

[40] Lei Chen, Zhengdao Chen, and Joan Bruna. On graph neural networks versus graph-augmented mlps. In *ICLR*. OpenReview.net, 2021.

[41] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[42] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *NeurIPS*, pages 14837–14847, 2019.

[43] Teague Sterling and John J. Irwin. ZINC 15 - ligand discovery for everyone. *J. Chem. Inf. Model.*, 55(11):2324–2337, 2015.

[44] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

[45] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *ICML*, Proceedings of Machine Learning Research, pages 12121–12132, 2021.

[46] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. Adversarial graph augmentation to improve graph contrastive learning. In *NeurIPS*, pages 15920–15933, 2021.

[47] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *ICLR*, 2022.

[48] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. Self-supervised graph-level representation learning with local and global structure. In *ICML*, volume 139, pages 11548–11558, 2021.

[49] Sihang Li, Xiang Wang, An Zhang, Yingxin Wu, Xiangnan He, and Tat-Seng Chua. Let invariant rationale discovery inspire graph contrastive learning. In *ICML*, pages 13052–13065, 2022.

[50] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. S2GAE: self-supervised graph autoencoders are generalizable learners with graph masking. In *WSDM*, pages 787–795. ACM, 2023.

[51] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *WWW*, volume abs/2304.04779, 2023.

[52] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. Pre-training molecular graph representation with 3d geometry. In *ICLR*. OpenReview.net, 2022.

[53] Thin Nguyen, Hang Le, Thomas P. Quinn, Tri Nguyen, Thuc Duy Le, and Svetha Venkatesh. Graphdta: predicting drug-target binding affinity with graph neural networks. *Bioinform.*, 37(8):1140–1147, 2021.

[54] Simon Axelrod and Rafael Gómez-Bombarelli. GEOM: energy-annotated molecular conformations for property prediction and molecular generation. *CoRR*, abs/2006.05531, 2020.

[55] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5):058301, 2012.

[56] Raghunathan Ramakrishnan, Mia Hartmann, Enrico Tapavicza, and O Anatole Von Lilienfeld. Electronic spectra from tddft and machine learning in chemical space. *The Journal of chemical physics*, 143(8), 2015.

[57] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[58] Quantum machine. http://quantum-machine.org/datasets/. Accessed 2023-03.

[59] Xiaomin Fang, Lihang Liu, Jieqiong Lei, Donglong He, Shanzhuo Zhang, Jingbo Zhou, Fan Wang, Hua Wu, and Haifeng Wang. Geometry-enhanced molecular representation learning for property prediction. *Nat. Mach. Intell.*, 4(2):127–134, 2022.

[60] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael M. Bronstein, and Federico Monti. SIGN: scalable inception graph neural networks. *CoRR*, abs/2004.11198, 2020.

[61] Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 4839–4848. PMLR, 2020.

[62] Shichang Zhang, Ziniu Hu, Arjun Subramonian, and Yizhou Sun. Motif-driven contrastive learning of graph representations. *arXiv preprint arXiv:2012.12533*, 2020.

[63] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[64] Steven Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, 2006.

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[66] Po-Yao Huang, Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, and Christoph Feichtenhofer. Masked autoencoders that listen. In *NeurIPS*, 2022.

[67] Alan Baade, Puyuan Peng, and David Harwath. MAE-AST: masked autoencoding audio spectrogram transformer. In *INTERSPEECH*, pages 2438–2442. ISCA, 2022.

[68] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*. OpenReview.net, 2021.

[69] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[70] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL (1)*. The Association for Computer Linguistics, 2016.

[71] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *NeurIPS*, 2020.

[72] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

[73] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *ICLR*. OpenReview.net, 2022.

[74] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *ICLR*. OpenReview.net, 2022.

[75] Ziqiao Zhang, Jihong Guan, and Shuigeng Zhou. Fragat: a fragment-oriented multi-scale graph attention model for molecular property prediction. *Bioinform.*, 37(18):2981–2987, 2021.

[76] Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. In *NeurIPS*, 2022.

[77] Chendi Qian, Gaurav Rattan, Floris Geerts, Mathias Niepert, and Christopher Morris. Ordered subgraph aggregation networks. In *NeurIPS*, 2022.

[78] Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests. *arXiv preprint arXiv:2302.07090*, 2023.

[79] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020.

[80] 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*, 2020.

[81] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, pages 3538–3545. AAAI Press, 2018.

[82] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *ICLR*. OpenReview.net, 2021.

[83] Yoav Levine, Barak Lenz, Opher Lieber, Omri Abend, Kevin Leyton-Brown, Moshe Tennenholtz, and Yoav Shoham. Pmi-masking: Principled masking of correlated spans. In *ICLR*. OpenReview.net, 2021.