Towards Effective and Generalizable Fine-tuning for Pre-trained Molecular Graph Models

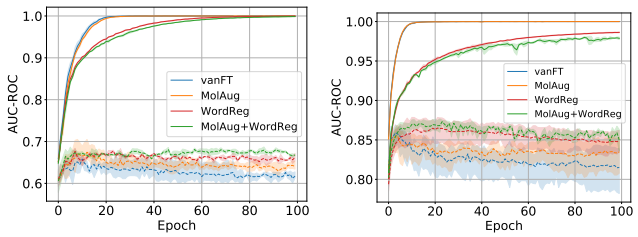
Jun Xia[1,2,3] , **Jiangbin Zheng**[2,3] , **Cheng Tan**[2,3] , **Ge Wang**[2,3] and **Stan Z. Li**[2,3]
[1]Zhejiang University [2]School of Engineering, Westlake University
[3]Institute of Advanced Technology, Westlake Institute for Advanced Study
{xiajun, zhengjiangbin, tancheng, wangge, stan.zq.li}@westlake.edu.cn

Abstract

Graph Neural Networks (GNNs) and Transformer have emerged as dominant tools for AI-driven drug discovery. Many state-of-the-art methods first pre-train GNNs or the hybrid of GNNs and Transformer on a large molecular database and then fine-tune on downstream tasks. However, different from other domains such as computer vision (CV) or natural language processing (NLP), getting labels for molecular data of downstream tasks often requires resource-intensive wet-lab experiments. Besides, the pre-trained models are often of extremely high complexity with huge parameters. These often cause the fine-tuned model to over-fit the training data of downstream tasks and significantly deteriorate the performance. To alleviate these critical yet under-explored issues, we propose two straightforward yet effective strategies to attain better generalization performance: 1. MolAug, which enriches the molecular datasets of down-stream tasks with chemical homologies and enantiomers; 2. WordReg, which controls the complexity of the pre-trained models with a smoothness-inducing regularization built on dropout. Extensive experiments demonstrate that our proposed strategies achieve notable and consistent improvements over vanilla fine-tuning and yield multiple state-of-the-art results. Also, these strategies are model-agnostic and readily pluggable into fine-tuning of various pre-trained molecular graph models. We will release the code and the fine-tuned models.

1 Introduction

Pre-trained language models (PLMs) have foundamentally changed the landscape of natural language processing (NLP) [Devlin *et al.*, 2019], which have established new state-of-the-art results for a large variety of NLP tasks. Inspired by their proliferation, tremendous efforts have been devoted to molecular graph pre-training which can exploit abundant knowledge of unlabelled molecular in the database [Hu *et al.*, 2020]. For the pre-training stage, existing works train the encoder with various pretext tasks in absence of labels [Rong *et al.*, 2020]. For the second fine-tuning stage, researchers adapt the pre-trained models to the downstream tasks via replacing the top layer of the pre-trained models by a task specific sub-network, and then continuing to train the new model with the limited data of the downstream task.



(a) SIDER (b) Tox21

Figure 1: Training (solid lines) and testing (dashed lines) curves of various fine-tuning strategies on SIDER and Tox21 datasets. 'vanFT' refers to vanilla fine-tuning. The over-fitting issue of vanilla fine-tuning impede the performance improvements while our proposed MolAug, WordReg or their combination can alleviate this issue.

Despite the fruitful progress in the strategies for pre-training, the fine-tuning stage remains under-explored for pre-trained molecular graph models. There are two crucial issues impede the improvements of performance during fine-tuning: (1) insufficient labeled data for downstream tasks. Different from other domains that have abundant labeled data, getting high-quality labels for molecular data often requires resource-intensive wet-lab experiments [Xia *et al.*, 2021a]. (2) the pre-trained models are often of extremely high complexity with tens millions of parameters [Rong *et al.*, 2020; Li *et al.*, 2021b], which posses the capability of memorizing the limited samples and lead to poor generalization [Mohri *et al.*, 2012; Haoming *et al.*, 2020]. As shown in Figure 1, we conduct vanilla fine-tuning on one of the state-of-the-art pre-trained molecular models MPG [Li *et al.*, 2021b]. The over-fitting issue poses hurdle to the further improvements on various datasets. To mitigate this issue, existing fine-tuning methods in other domains often rely on hyper-parameter tuning heuristics. For example, Howard and Ruder [Howard and Ruder, 2018] follow a heuristic learning rate schedule and gradually unfreeze the layers of the pre-trained language model to improve the fine-tuning performance, which require significant tuning efforts. The other line of works propose various regularizations to control complexity of pre-trained models. Specifically, SMART [Haoming *et al.*, 2020] intro-

duces a regularization, which encourages the output of the model not to change much, when injecting an adversarial perturbation to the input. However, this strategy is not suitable for pre-trained graph models because synthesizing adversarial samples for molecular graphs online is time-consuming for its high complexity. Recently, Child-Tuning [Xu *et al.*, 2021b] is proposed to strategically mask out some gradients of network during the backward process. Albeit effective, it suffers a heavy computational overhead because it requires to search for suitable masks for each parameter in the pre-trained models. Now, we are naturally motivated to ask following question: *Can we devise more suitable strategies for effective and generalizable fine-tuning on pre-trained molecular graph models?*

To fully harness the power of pre-trained molecular graph models, we propose two strategies for better fine-tuning: MolAug and WordReg. MolAug is molecular graph augmentation with chemical enantiomers and homologies which share the similar (or identical) physical (permeability, solubility, etc.) or chemical (toxicity, etc.) properties with themselves. WordReg is a novel smoothness-inducing regularization built on dropout. To the best of our knowledge, we are the first to study the fine-tuning stage of the pre-trained molecular graph models, which is important while being neglected. We summarize our contributions here:

- We propose new data augmentations for molecular graph data, which introduce variations while not altering the physical or chemical properties of molecules too much.
- We propose a novel smoothness-inducing regularization built on dropout that can effectively control the high complexity of the pre-trained models. Also, it is domain-agnostic and we study its effects in fine-tuning pre-trained models of NLP and CV in the appendix.
- Through extensive experiments on 3 drug discovery subtasks (11 datasets in total) including molecular property prediction, drug-drug interaction prediction and drug-target interaction prediction, we observe consistent and notable improvements over vanilla fine-tuning and yield multiple new state-of-the-art results.

2 Related work

2.1 Molecular Representation Learning

Molecular Representation Learning refers to represent molecules in the vector space. Initially, the traditional chemical fingerprints such as ECFP [Rogers and Hahn, 2010] encode the neighbors of atoms in the molecule into a fix-length vector. With the proliferation of deep learning, [Duvenaud *et al.*, 2015] first introduce convolutional neural networks to learn neural fingerprints of molecules. Subsequently, some researchers feed the SMILES (a line notation for describing the structure of chemical species using short ASCII strings) into RNN-based models to obtain molecular representations. In order to fulfill the topology information of molecular graphs, some recent works [Kearnes *et al.*, 2016; Xiong *et al.*, 2020] attempt to apply GNNs to molecular representation learning. Besides, MPNN [Gilmer and others., 2017] and its variants DMPNN [Yang *et al.*, 2019], CMPNN [Song *et al.*, 2020], CoMPT [Chen *et al.*, 2021] utilize a message passing framework to better capture the interactions among atoms. However, they still require expensive annotations and barely generalize to unseen molecules, which pose hurdle to the practical applications.

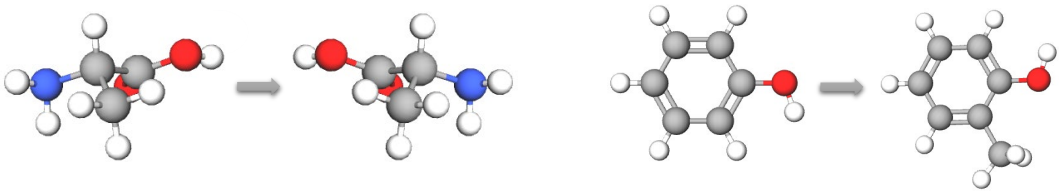
2.2 Pre-training on Graphs

To address the fundamental challenges of extremely scarce labeled data and out-of-distribution generalization in graphs learning, tremendous efforts have been devoted to pre-training on graphs recently. One line of these works follow the contrastive paradigm [Zhu *et al.*, 2021b; Zhu *et al.*, 2021a; Xia *et al.*, 2021b]. For molecular pre-training, GraphCL [You *et al.*, 2020] and its variants [You *et al.*, 2021; Susheel *et al.*, 2021; Xia *et al.*, 2022] embeds augmented versions of the anchor molecular graph close to each other and pushes the embeddings of other molecules apart. The other line of works adopt generative pretext tasks. Prototypical examples are GPT-GNN [Hu and others., 2020] which introduces a self-supervised attributed graph generation task to pre-train GNNs so that they can capture the structural and semantic properties of the graph. For molecular graph pre-training, Hu et al. [Hu *et al.*, 2020] conduct attribute and structure prediction at the level of individual nodes as well as entire graphs. To capture the rich information in molecular graph motifs, GROVER [Rong *et al.*, 2020] and MGSSL [Zhang *et al.*, 2021] propose to predict or generate the motifs. Analogously, MPG [Li *et al.*, 2021b] learns to compare two half-graphs (each decomposed from a graph sample) and discriminate whether they come from the same source. Despite the progress in molecular graph pre-training, few efforts have been devoted to the fine-tuning except for a recent work [Han and others., 2021] that adaptively selects and combines various auxiliary tasks with the target task in the fine-tuning stage to improve performance, which is impractical because auxiliary tasks are often unavailable during fine-tuning.

3 Methodology

3.1 MolAug: Molecular Graph Augmentations with Chemical Enantiomers and Homologies

Initially, GraphCL [You *et al.*, 2020] augments molecular graph data in the form of naive random corruption (e.g., dropping bonds, dropping atoms and etc.). However, these augmentations may alter molecular graph semantics completely even if the perturbation is weak. For example, dropping a carbon atom in the phenyl ring will alter the aromatic system and result in an alkene chain, which will drastically change the molecular properties. Besides, MoCL [Sun *et al.*, 2021] attempts to incorporate domain knowledge into graph data augmentation via replacing valid substructures in molecular graph with bioisosteres that share similar properties. However, bioisosteres are used to modify some molecular properties as expected (e.g, reduce toxicity) in drug design [Mannhold *et al.*, 2012], which may introduce incorrect supervision for molecular properties prediction such as toxicity prediction. We compare MolAug with these augmentations in section 4.5.



(a) L-alanine (b) D-alanine (c) Phenol (d) 2-methylphenol

Figure 2: Illustration of MolAug. L-alanine (a) and D-alanine (b) is a pair of enantiomers; Phenol and 2-methylphenol is a pair of homologies. Gray, red, blue and white balls denote Carbon (C), Oxygen (O), Nitrogen (N) and Hydrogen (H) atoms respectively.

To alleviate these issues, we resort to chemical enantiomers and homologies. As shown in Figure 2 (a) and (b), an enantiomer is one of two stereoisomers that are mirror images of each other that are non-superposable, much as one's left and right hands are mirror images of each other that cannot appear identical simply by reorientation. Despite the structural difference, enantiomers share the identical chemical and physical properties with each other in most cases and thus being a more suitable alternative for molecular graph augmentations in molecular properties prediction. In practice, we can obtain chemical enantiomers of a molecule by changing the chirality type which is provided in the atom (node) feature. Besides, as shown in Figure 2 (c) and (d), chemical homologies are a series of compounds differing from each other by a repeating unit, such as a methylene bridge, which share the same chemical properties with each other. Therefore, chemical homologies can serve as an ideal way of augmentation when predicting molecular chemical properties. For each molecule $\mathbf{x}$ and its label y in the dataset $\mathcal{D}$ of down-stream task, we can obtain its various augmented versions with operation $\mathrm{MolAug}(\cdot)$ which can be enantiomers, homologies or their combinations. Now, we can formulate the loss of fine-tuning with MolAug as,

$$\mathcal{L}_M = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}}(l\left(y, f_{\boldsymbol{\theta}}\left(\mathbf{x}\right)\right) + \mu_1 l\left(y, f_{\boldsymbol{\theta}}(\mathrm{MolAug}(\mathbf{x}))\right)), \tag{1}$$

where μ_1 is a trade-off parameter controlling the impact of MolAug and l is the loss function of downstream task. We can also obtain various augmentations for each molecule during each iterations with MolAug, which we study in section 4.6. Apart from pretrain-then-finetune paradigm, MolAug can also work well in training from scratch, which we validate in the appendix.

3.2 WordReg: Smoothness-inducing Regularization Built on Dropout

Existing fine-tuning strategies [Haoming *et al.*, 2020; Xu *et al.*, 2021b; Aghajanyan *et al.*, 2021] in NLP enforce the output of the model not to change much, when injecting perturbations to the input or model parameters. In other words, they encourage the model to be insensitive to perturbations, and thus effectively control its capacity [Mohri *et al.*, 2012]. Compared with these strategies, WordReg imposes the perturbations on the neurons which is more efficient than perturbing the input or huge model parameters. Besides, as shown in

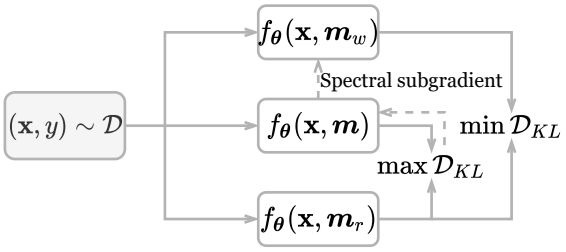


Figure 3: Illustration of WordReg. We obtain the worst-case drop mask vector $\boldsymbol{m}_w$ (corresponding to $\boldsymbol{m}_r$) via solving a BQP problem with spectral subgradient-based method.

Figure 3, WordReg obtains the worst-case dropout via maximizing the divergence between the outputs with two different dropout which is task-dependent and can introduce stronger regularization than random dropout (validated in Figure 6). Formally, consider the pre-trained neural networks $f_\theta(\mathbf{x}, \boldsymbol{m})$ that takes sample $\mathbf{x}$ as input when fine-tuning on downstream task. $\boldsymbol{m}$ is a mask vector denoting which neuron of the pre-trained model should be dropped. More specifically, for the i-th unit m_i of vector $\boldsymbol{m}$, $m_i = 1$ indicates that the neuron should be dropped while $m_i = 0$ illustrates that the neuron should be preserved during dropout. To start, we introduce $\boldsymbol{m}_r$ to denote the mask vector of random dropout of vanilla training. With the dropout ratio $\sigma \in [0, 1]$ preseted, we can define the constraint on $\boldsymbol{m}$ as,

$$\mathcal{R}_{\boldsymbol{m}} = \left\{\boldsymbol{m} \mid \boldsymbol{m} \in \{0,1\}^N, \|\boldsymbol{m}\|_0 = \lfloor \sigma N \rfloor\right\}, \tag{2}$$

where $\|\cdot\|_0$ is the ℓ_0 norm, N is the number of neurons in the model. Therefore, we can obtain the worst-case mask vector $\boldsymbol{m}_w$ by solving an optimization problem,

$$\boldsymbol{m}_w = \mathop{\arg\max}_{\boldsymbol{m}\in\mathcal{R}_{\boldsymbol{m}}} \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m})\right), \tag{3}$$

where $\mathcal{D}_{KL}$ is Kullback–Leibler divergence. With Taylor expansion, we can approximate the optimal solution as,

$$\boldsymbol{m}_w \approx \mathop{\arg\max}_{\boldsymbol{m}\in\mathcal{R}_{\boldsymbol{m}}} \frac{1}{2}\boldsymbol{m}^T \boldsymbol{H} \boldsymbol{m}, \tag{4}$$

$$\boldsymbol{H} = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \nabla^2 \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m})\right)\Big|_{\boldsymbol{m}=\mathbf{0}}, \tag{5}$$

$\boldsymbol{H}$ is the Hessian matrix of the loss $\mathcal{D}_{KL}$ at $\boldsymbol{m} = \mathbf{0}$ which is a $N \times N$ semi-positive definite matrix. Obviously, this is a Binary Quadratic Programming (BQP) problem, which is NP-hard but admits an approximate solution to

$\boldsymbol{m}_w$. Here, we propose a novel method to solve this problem efficiently based on the spectral subgradient [Boyd *et al.*, 2003]. Firstly, we convert $\{0,1\}$-constraint on $\boldsymbol{m}$ of $\mathcal{R}_{\boldsymbol{m}}$ to $\{-1,1\}$-constraint on $\boldsymbol{n}$ via defining $\boldsymbol{n} = 2\boldsymbol{m} - 1$. Then we introduce a new variable $\widehat{\boldsymbol{n}}$ with $N+1$ dimensions,

$$\widehat{\boldsymbol{n}}_i = \begin{cases} 1, & \text{if} \quad i = N+1; \\ \boldsymbol{n}_i, & \text{otherwise.} \end{cases} \tag{6}$$

Then we can rewrite constraint $\mathcal{R}_{\boldsymbol{m}}$ as a new constraint $\mathcal{R}_{\widehat{\boldsymbol{n}}}$ on $\widehat{\boldsymbol{n}}$,

$$\mathcal{R}_{\widehat{\boldsymbol{n}}} = \left\{ \widehat{\boldsymbol{n}} \mid \widehat{\boldsymbol{n}} \in \{\pm 1\}^{N+1}, \boldsymbol{e}^T \widehat{\boldsymbol{n}} = c \right\}, \tag{7}$$

where $c = 2\lfloor \sigma N \rfloor - N + 1$ and $\boldsymbol{e} \in \mathbb{R}^{N+1}$ is an all-one vector. By these transformations, we can reformulate the BQP in Eq (5) as a new BQP in terms of $\widehat{\boldsymbol{n}}$, where the constraint term $\widehat{\boldsymbol{n}} \in \{\pm 1\}^{N+1}$ can be rewrited as $\widehat{\boldsymbol{n}}_i^2 = 1$. We then introduce a Lagrange multiplier λ_i for each constraint $\widehat{\boldsymbol{n}}_i^2 = 1$ and λ_0 for the constraint $\boldsymbol{e}^T \widehat{\boldsymbol{n}} = c$. Now, we can formulate the dual problem of the original BQP as,

$$\min_{\boldsymbol{\lambda}, \lambda_0} d\left(\boldsymbol{\lambda}, \lambda_0\right), \tag{8}$$

with

$$\begin{aligned} d\left(\boldsymbol{\lambda}, \lambda_0\right) &= \max_{\|\widehat{\boldsymbol{n}}\|^2 = N+1} \widehat{\boldsymbol{n}}^T [\boldsymbol{L} + \operatorname{diag}(\boldsymbol{\lambda})] \widehat{\boldsymbol{n}} - \boldsymbol{e}^T \boldsymbol{\lambda} - c\lambda_0 \\ &= (N+1)\lambda_{\max} - \boldsymbol{e}^T \boldsymbol{\lambda} - c\lambda_0 \end{aligned} \tag{9}$$

where

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{H} & \boldsymbol{H}\boldsymbol{e} + \frac{1}{2}\lambda_0 \boldsymbol{e} \\ \boldsymbol{e}^T \boldsymbol{H} + \frac{1}{2}\lambda_0 \boldsymbol{e}^T & 0 \end{pmatrix} \in \mathbb{R}^{N+1 \times N+1}$$

, and $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{L} + \operatorname{diag}(\boldsymbol{\lambda})$. The eigenvector of unit norm $\mathbf{u}_{\max}$ corresponding to $\lambda_{\max}$ can be derived via approximated by using a single-step power iteration instead of conducting naive eigenvalue decomposition. Then, the maximum $\widehat{\boldsymbol{n}}^*$ can be derived,

$$\widehat{\boldsymbol{n}}^* = \sqrt{N+1}\mathbf{u}_{\max}. \tag{10}$$

The dual problem Eq.(8) can be solved by the gradient descent method over iterations. We show the gradient of d w.r.t $\boldsymbol{\lambda}$ and λ_0 as follows,

$$\nabla_{\boldsymbol{\lambda}} d = (N+1)\mathbf{u}_{\max}^2 - \boldsymbol{e}, \tag{11}$$

$$\frac{\partial d}{\partial \lambda_0} = \frac{1}{2}(N+1)\mathbf{u}_{\max}^T \begin{pmatrix} 0 & \boldsymbol{e} \\ \boldsymbol{e} & 0 \end{pmatrix} \mathbf{u}_{\max} - c, \tag{12}$$

where $\mathbf{u}_{\max}^2$ denotes an element-wise square of $\mathbf{u}_{\max}$. During fine-tuning the pre-trained models with WordReg, over each mini-batch, we compute the above gradient to make an one-step update of the Lagrange multipliers $\boldsymbol{\lambda}$ and λ_0 along the descending direction, before the maximum $\widehat{\boldsymbol{n}}^*$ is taken with the updated multipliers. Finally, both $\pm\widehat{\boldsymbol{n}}^*$ are optimal and we should choose the one closer to $\widehat{\boldsymbol{n}}_{N+1} = 1$ as required. Note that we seek the worst-case dropout mask layer-by-layer instead of applying it to an entire network as a whole. This can make the WordReg computationally efficient as well as prevent too many neurons from being dropped at a few layers.

With the worst-case dropout mask vector $\boldsymbol{m}_w$ obtained, we can formulate the loss of fine-tuning with WordReg as,

$$\mathcal{L}_W = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} (l\left(y, f_{\boldsymbol{\theta}}\left(\mathbf{x}\right)\right) + \mu_2 \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r), f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_w)\right)), \tag{13}$$

where μ_2 is a trade-off parameter controlling the impact of WordReg, which we study in section 4.6.

Remark: There are also some regularizations ELD [Ma *et al.*, 2017] and FD [Zolna *et al.*, 2018] built on dropout. WordReg differs from them in two aspects: (1) WordReg introduces the worst-cast dropout, which is task-dependent and possess stronger regularization ability than their random drop; (2) ELD and FD are designed to reducing the gap between training (sub-model with dropout) and inference (full model without dropout) while WordReg controls the gap between sub-model with different dropouts.

3.3 Combining MolAug and WordReg

Additionally, we can conduct fine-tuning with MolAug and WordReg simultaneously. Formally, we specify the loss function of combining MolAug and WordReg in Eq (14).

$$\begin{aligned} \mathcal{L}_{M+W} = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} (&l\left(y, f_{\boldsymbol{\theta}}\left(\mathbf{x}\right)\right) + \mu_1 l\left(y, f_{\boldsymbol{\theta}}(\text{MolAug}(\mathbf{x}))\right) \\ &+ \mu_2 \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r), f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_w)\right)). \end{aligned} \tag{14}$$

The pseudo codes can be found in the appendix.

4 Experiments

4.1 Experimental Setup

Base Pre-trained Molecular Graph Models. We adopt recent proposed pre-trained molecular graph models MPG [Li *et al.*, 2021b] as the base model and fine-tune it with our proposed strategies. We also adopt MGSSL [Zhang *et al.*, 2021] and GraphLog [Xu *et al.*, 2021a] as our base models to valuate that our strategies can be plugged into the fine-tuning of various pre-trained molecular graph models.

Fine-tuning Tasks and Datasets. Following previous works, we use 8 benchmark datasets from the MoleculeNet [Wu *et al.*, 2018] to perform molecular property prediction. To keep fair, we adopt the scaffold splitting method with a ratio for train/validation/test as 8:1:1 as previous works. We apply a grid search procedure to obtain the best hyperparameters with validation set. More details of the hyperparameters and datasets are deferred to the appendix. *The experiments on drug-drug interaction prediction and drug-target interaction prediction can be found in the appendix.*

Baselines. We adopt two types of baselines:

- *Training from scratch:* TF_Roubust [Rogers and Hahn, 2010] is a DNN-based multitask framework taking the molecular fingerprints as the input. GCN [Kipf. and Welling, 2017], Weave [Kearnes *et al.*, 2016] and SchNet [Schuett *et al.*, 2017] are three graph convolutional models. MPNN and its variants DMPNN, MGCN, CMPNN and CoMPT have been introduced in the related work. AttentiveFP [Xiong *et al.*, 2020] is an extension of the graph attention network.

Table 1: The performance comparison (*higher is better for classification task, lower is better for regression task*). *Relative Improvement* refers to the absolute improvement over MPG base model divided by the original results of MPG. We adopt it as a unified description of improvements for both classification and regression. The methods marked with '*' means the original papers follow a different data splitting with MPG, we fine-tune their public pre-trained models with the splitting as MPG and report the results here. The numbers in brackets are the standard deviation and the ones underlined are the previous best results. '—' means the methods are too time-consuming or original implementations do not admit regression task without non-trivial modifications.

Task	Classification (ROC-AUC)						Regression (RMSE)	
Dataset	BBBP	SIDER	ClinTox	BACE	Tox21	ToxCast	FreeSolv	ESOL
# Molecules	2039	1427	1478	1513	7831	8575	642	1128
ECFP [Rogers and Hahn, 2010]	$0.783_{(0.050)}$	$0.630_{(0.019)}$	$0.673_{(0.031)}$	$0.861_{(0.024)}$	$0.760_{(0.009)}$	$0.615_{(0.017)}$	$5.275_{(0.751)}$	$2.359_{(0.454)}$
TF_Robust [Ramsundar and others., 2015]	$0.860_{(0.087)}$	$0.607_{(0.033)}$	$0.765_{(0.085)}$	$0.824_{(0.022)}$	$0.698_{(0.012)}$	$0.585_{(0.031)}$	$4.122_{(0.085)}$	$1.722_{(0.038)}$
GraphConv [Kipf. and Welling, 2017]	$0.877_{(0.036)}$	$0.593_{(0.035)}$	$0.845_{(0.051)}$	$0.854_{(0.011)}$	$0.772_{(0.041)}$	$0.650_{(0.025)}$	$2.900_{(0.135)}$	$1.068_{(0.050)}$
Weave [Kearnes *et al.*, 2016]	$0.837_{(0.065)}$	$0.543_{(0.034)}$	$0.823_{(0.023)}$	$0.791_{(0.008)}$	$0.741_{(0.044)}$	$0.678_{(0.024)}$	$2.398_{(0.250)}$	$1.158_{(0.055)}$
SchNet [Schuett *et al.*, 2017]	$0.847_{(0.024)}$	$0.545_{(0.038)}$	$0.717_{(0.042)}$	$0.750_{(0.033)}$	$0.767_{(0.025)}$	$0.679_{(0.021)}$	$3.215_{(0.755)}$	$1.045_{(0.064)}$
MPNN [Gilmer and others., 2017]	$0.913_{(0.041)}$	$0.595_{(0.030)}$	$0.879_{(0.054)}$	$0.815_{(0.044)}$	$0.808_{(0.024)}$	$0.691_{(0.013)}$	$2.185_{(0.952)}$	$1.167_{(0.430)}$
DMPNN [Yang *et al.*, 2019]	$0.919_{(0.030)}$	$0.632_{(0.023)}$	$0.897_{(0.040)}$	$0.852_{(0.053)}$	$0.826_{(0.023)}$	$0.718_{(0.011)}$	$2.177_{(0.914)}$	$0.980_{(0.258)}$
MGCN [Lu *et al.*, 2019]	$0.850_{(0.064)}$	$0.552_{(0.018)}$	$0.634_{(0.042)}$	$0.734_{(0.030)}$	$0.707_{(0.016)}$	$0.663_{(0.009)}$	$3.349_{(0.097)}$	$1.266_{(0.147)}$
AttentiveFP [Xiong *et al.*, 2020]	$0.908_{(0.050)}$	$0.605_{(0.060)}$	$0.933_{(0.020)}$	$0.863_{(0.015)}$	$0.807_{(0.020)}$	$0.579_{(0.001)}$	$2.030_{(0.420)}$	$0.853_{(0.060)}$
TrimNet [Li *et al.*, 2021a]	$0.892_{(0.025)}$	$0.606_{(0.006)}$	$0.906_{(0.017)}$	$0.843_{(0.025)}$	$0.812_{(0.019)}$	$0.652_{(0.032)}$	$2.529_{(0.111)}$	$1.282_{(0.029)}$
CMPNN [Song *et al.*, 2020]	$0.927_{(0.017)}$	$0.616_{(0.003)}$	$0.902_{(0.008)}$	$0.856_{(0.012)}$	$0.806_{(0.016)}$	$0.681_{(0.008)}$	$2.007_{(0.442)}$	$0.798_{(0.112)}$
CoMPT [Chen *et al.*, 2021]	$0.938_{(0.021)}$	$0.634_{(0.030)}$	$0.934_{(0.019)}$	$0.859_{(0.016)}$	$0.817_{(0.014)}$	$0.693_{(0.025)}$	$1.855_{(0.578)}$	$0.774_{(0.058)}$
Mol2Vec [Jaeger *et al.*, 2018]	$0.876_{(0.030)}$	$0.601_{(0.023)}$	$0.828_{(0.023)}$	$0.841_{(0.052)}$	$0.805_{(0.015)}$	$0.690_{(0.014)}$	$5.752_{(1.245)}$	$2.358_{(0.452)}$
N-GRAM [Liu *et al.*, 2019]	$0.912_{(0.013)}$	$0.632_{(0.005)}$	$0.855_{(0.037)}$	$0.876_{(0.035)}$	$0.769_{(0.027)}$	—	$2.512_{(0.190)}$	$1.100_{(0.160)}$
SMILES-BERT [Wang *et al.*, 2019]	$\underline{0.959}_{(0.009)}$	$0.568_{(0.031)}$	$\underline{0.985}_{(0.014)}$	$0.849_{(0.021)}$	$0.803_{(0.010)}$	—	$2.974_{(0.510)}$	$0.841_{(0.096)}$
HU.et.al.* [Hu *et al.*, 2020]	$0.915_{(0.040)}$	$0.614_{(0.006)}$	$0.762_{(0.058)}$	$0.851_{(0.027)}$	$0.811_{(0.015)}$	$0.714_{(0.019)}$	—	—
GraphCL* [You *et al.*, 2020]	$0.906_{(0.028)}$	$0.629_{(0.037)}$	$0.865_{(0.017)}$	$0.872_{(0.023)}$	$0.826_{(0.037)}$	$0.728_{(0.025)}$	—	—
JOAO* [You *et al.*, 2021]	$0.895_{(0.033)}$	$0.634_{(0.011)}$	$0.869_{(0.022)}$	$0.863_{(0.021)}$	$0.818_{(0.021)}$	$0.733_{(0.006)}$	—	—
AD-GCL* [Susheel *et al.*, 2021]	$0.918_{(0.025)}$	$0.651_{(0.020)}$	$0.884_{(0.032)}$	$0.881_{(0.017)}$	$0.826_{(0.031)}$	$0.742_{(0.012)}$	—	—
GraphLog* [Xu *et al.*, 2021a]	$0.902_{(0.007)}$	$0.648_{(0.015)}$	$0.875_{(0.009)}$	$0.873_{(0.022)}$	$0.823_{(0.017)}$	$0.738_{(0.0021)}$	—	—
MGSSL* [Zhang *et al.*, 2021]	$0.933_{(0.009)}$	$0.656_{(0.011)}$	$0.906_{(0.019)}$	$0.889_{(0.006)}$	$0.828_{(0.015)}$	$0.740_{(0.018)}$	—	—
GROVER [Rong *et al.*, 2020]	$0.940_{(0.019)}$	$0.658_{(0.023)}$	$0.944_{(0.021)}$	$0.894_{(0.028)}$	$0.831_{(0.025)}$	$0.737_{(0.010)}$	$1.544_{(0.397)}$	$0.831_{(0.120)}$
MPG [Li *et al.*, 2021b]	$0.922_{(0.012)}$	$\underline{0.661}_{(0.007)}$	$0.963_{(0.028)}$	$\underline{0.920}_{(0.013)}$	$\underline{0.837}_{(0.019)}$	$\underline{0.748}_{(0.005)}$	$\underline{1.269}_{(0.192)}$	$\underline{0.741}_{(0.017)}$
MPG + MolAug	$0.958_{(0.022)}$	$0.672_{(0.025)}$	$0.968_{(0.017)}$	$0.928_{(0.07)}$	$0.849_{(0.022)}$	$0.760_{(0.010)}$	$0.976_{(0.028)}$	$0.699_{(0.052)}$
MPG + WordReg	$0.975_{(0.015)}$	$0.676_{(0.010)}$	$\mathbf{0.991}_{(0.012)}$	$\mathbf{0.946}_{(0.014)}$	$0.857_{(0.010)}$	$0.762_{(0.010)}$	$0.988_{(0.130)}$	$\mathbf{0.664}_{(0.048)}$
MPG + MolAug + WordReg	$\mathbf{0.978}_{(0.016)}$	$\mathbf{0.685}_{(0.012)}$	$0.986_{(0.021)}$	$0.932_{(0.016)}$	$\mathbf{0.862}_{(0.011)}$	$\mathbf{0.763}_{(0.008)}$	$\mathbf{0.934}_{(0.093)}$	$0.695_{(0.039)}$
Relative Improvement	+6.1%	+3.6%	+2.9%	+2.8%	+3.0%	+2.0%	+26.4%	+10.4%

- *Pretrain-then-finetune:* We also compare with several pre-trained models with vanilla fine-tuning: Mol2Vec [Jaeger *et al.*, 2018], N-Gram [Liu *et al.*, 2019], SMILES-BERT [Wang *et al.*, 2019], GROVER, Hu et.al, MPG and several graph contrastive learning methods as introduced in the related work.

4.2 Comparisons with State-of-the-art Results

We first fine-tune the state-of-the-art pre-trained molecular graph model MPG with MolAug, WordReg and their combinations respectively. The results are reported in Table 1 which provides the following observations: (1) MolAug and WordReg consistently perform better than vanilla fine-tuning on MPG. The overall relative improvement is 7.2% (3.4% on classification tasks and 18.4% on regression tasks) ; (2) Equipped with MolAug, WordReg or their combinations, MPG yields new state-of-the-art results on molecular property prediction; (3) The combination of MolAug and WordReg brings a 26.4% relative improvement over MPG on a small dataset FreeSolv with only 642 labeled molecules. This confirms that our strategies can alleviate the over-fitting issues which is often severer in small-scaled datasets.

4.3 Comparisons with Other Fine-tuning Methods

Due to the fact that there is no fine-tuning methods specially designed for pre-trained molecular graph models, we consider some start-of-the-arts fine-tuning strategies in other domains. As can be observed in Table 2, generally, existing fine-tuning techniques can also bring improvements for Pre-trained molecular graph models. Compared with them, MolAug and WordReg are more suitable for pre-trained molecular graph models because they can avoid heavy tunning or heavy computational overhead of other fine-tuning methods and achieve the best performance among them.

Table 2: Comparison with other fine-tuning methods. The running time is evaluated on the same device (Tesla V100 GPU). We provide more details and results in the appendix.

Methods	BBBP		SIDER	
	ROC-AUC	Time (s)	ROC-AUC	Time (s)
Vanilla Fine-tuning (Base model)	$0.922_{(0.012)}$	$654.6_{(7.9)}$	$0.661_{(0.007)}$	$327.6_{(5.2)}$
SMART [Haoming *et al.*, 2020]	$0.959_{(0.009)}$	$9583.3_{(26.2)}$	$0.673_{(0.005)}$	$8927.5_{(31.1)}$
R3F [Aghajanyan *et al.*, 2021]	$0.931_{(0.008)}$	$3876.5_{(35.6)}$	$0.659_{(0.013)}$	$2930.5_{(16.7)}$
Adaptive Fine-tuning [Han and others., 2021]	$0.944_{(0.012)}$	$8051.2_{(8.2)}$	$0.653_{(0.012)}$	$6982.3_{(15.9)}$
ChildTuning [Xu *et al.*, 2021b]	$0.952_{(0.005)}$	$8541.4_{(24.8)}$	$0.668_{(0.017)}$	$7358.6_{(22.8)}$
MolAug	$0.958_{(0.027)}$	$789.8_{(15.1)}$	$0.672_{(0.019)}$	$405.6_{(5.7)}$
WordReg	$\underline{0.975}_{(0.015)}$	$1659.8_{(10.1)}$	$\underline{0.676}_{(0.010)}$	$1436.3_{(20.3)}$
MolAug + WordReg	$\mathbf{0.978}_{(0.016)}$	$2322.4_{(16.8)}$	$\mathbf{0.685}_{(0.012)}$	$1791.68_{(14.9)}$

4.4 Improving the Performance of Various Pre-trained Molecular Graph Models

To validate that MolAug and WordReg are model-agnostic, we fine-tune other pre-trained molecular graph models including MGSSL and GraphLog (see in the appendix for the limited space) with our MolAug and WordReg. Figure 4 shows the comparisons between our strategies and vanilla fine-tuning on the pre-trained model MGSSL. Obviously, our strategies outperform vanilla fine-tuning across various datasets, which validates that MolAug & WordReg

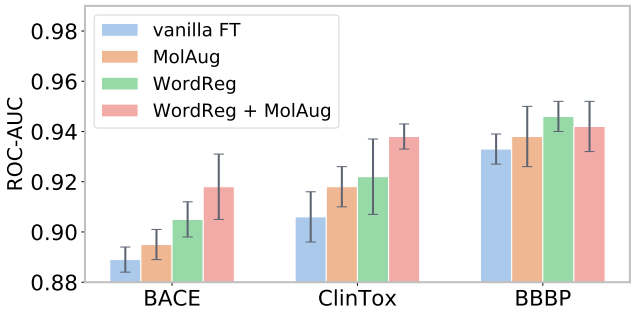
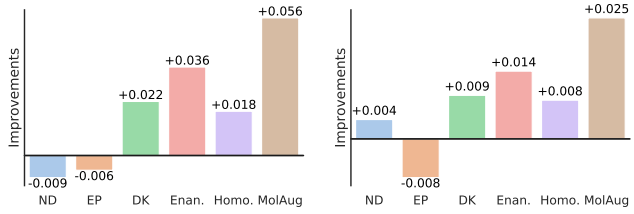


Figure 4: MolAug and WordReg's performance (compared with vanilla fine-tuning) on another pre-trained model MGSSL.

can also improve the performance when fine-tuning various pre-trained molecular graph models.

4.5 Ablation Study

Ablation Study for MolAug. In this section, we substitute MolAug with general graph augmentations including nodes drop (ND), edges perturbation (EP) in GraphCL [You *et al.*, 2020] and domain knowledge-enriched molecular graph augmentations (DK) in MoCL [Sun *et al.*, 2021]. Besides, we also consider chemical enantiomers (Enan.) and homologies (Homo.) of MolAug in isolation. The results demonstrated in Figure 5 illustrate that general graph augmentations and DK bring negative or limited improvements over vanilla fine-tuning, which may alter molecular property completely. Besides, using Enan. and Homo. in isolation is not the optimal alternative. Instead, the combinations of these two ingredients of MolAug achieve the best performance.



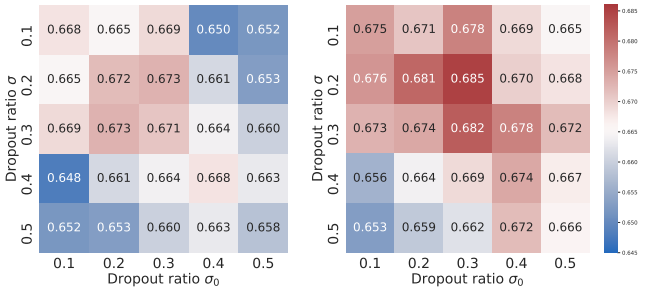
(a) BBBP (b) Tox21

Figure 5: Comparisons between MolAug and existing molecular graph augmentations. The improvements refer to the improvements over vanilla fine-tuning.

Ablation Study for WordReg. We substitute the worst-case dropout with random dropout to study its influence. As shown in Figure 6, WordReg outperforms the random dropout across various drop ratios, which validates that looking for the worst-case dropout is necessary and effective. Figure 6(a) is symmetrical because both two dropouts are random.

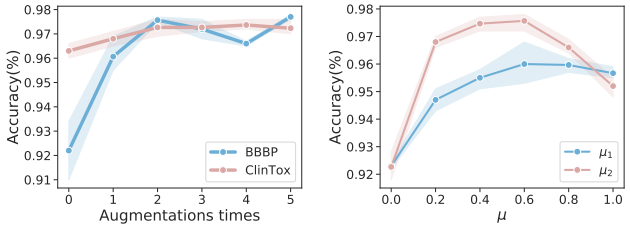
4.6 Hyper-parameters Analysis

Worst-case dropout ratio. We have study dropout ratio in Figure 6(b). Obviously, WordReg can achieve brilliant performance when the two dropout ratios are in a reasonable



(a) Random (b) Worst-case

Figure 6: Comparisons between WordReg and random dropout regularization. The experiments are conducted with MPG base model on SIDER dataset. Here σ_0 and σ are dropout ratio of vanilla training (fine-tuning) and worst-case drop, respectively.



(a) Augmentations Times (b) Trade-off Parameters

Figure 7: Hyper-parameters Analysis.

range (0.1-0.3). Besides, WordReg tends to perform better when the two dropout ratios are close or identical.

Augmentations times of MolAug. As shown in Figure 7(a), more augmentations for each molecule at each iteration bring improvements. However, the performance is prone to be steady when the augmentations times are up to 3. On the other hand, more augmentations times will lead to unnecessary computational overhead.

Trade-off parameters μ_1, μ_2. Figure 7(b) shows an different influence between μ_1 and μ_2. For μ_1, the accuracy will be improved and then tend to converge with its value increasing, which illustrates that MolAug can also work well even if its trade-off parameter is oversized. In contrast, for μ_2, the accuracy sees a dramatical drop, which can be imputed to its over-powerful regularization.

5 Conclusions and Future Work

We propose two effective strategies, MolAug & WordReg, for fine-tuning pre-trained molecular graph models to alleviate the over-fitting issues. The empirical results suggest that our strategies can advance the performance of vanilla fine-tuning on various pre-trained models and outperform state-of-the-arts results by a large margin. Also, we validate that our strategies are superior to existing fine-tuning techniques. Interesting direction for future work is applying our strategies to other molecule-related tasks such as molecule generation.

References

[Aghajanyan *et al.*, 2021] Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. Better fine-tuning by reducing representational collapse. In *ICLR*, 2021.

[Boyd *et al.*, 2003] Stephen Boyd, Lin Xiao, and others. Subgradient methods. 2003.

[Chen *et al.*, 2021] Jianwen Chen, Shuangjia Zheng, and others. Learning attributed graph representation with communicative message passing transformer. *IJCAI*, 2021.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, and others. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.

[Duvenaud *et al.*, 2015] David Duvenaud, Dougal Maclaurin, and others. Convolutional networks on graphs for learning molecular fingerprints. *NIPS*, 2015.

[Gilmer and others., 2017] Justin Gilmer and others. Neural message passing for quantum chemistry. *ICML*, 2017.

[Han and others., 2021] Xueting Han and others. Adaptive transfer learning on graph neural networks. In *KDD*, 2021.

[Haoming *et al.*, 2020] Jiang Haoming, He Pengcheng, and . others. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *ACL*, 2020.

[Howard and Ruder, 2018] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *ACL*, 2018.

[Hu and others., 2020] Ziniu Hu and others. Gpt-gnn: Generative pre-training of graph neural networks. *KDD*, 2020.

[Hu *et al.*, 2020] Weihua Hu, Bowen Liu, and others. Strategies for pre-training graph neural networks. *ICLR*, 2020.

[Jaeger *et al.*, 2018] Sabrina Jaeger, Simone Fulle, and others. Mol2vec: Unsupervised machine learning approach with chemical intuition. *J CHEM INF MODEL*, 2018.

[Kearnes *et al.*, 2016] Steven Kearnes, Kevin McCloskey, and others. Molecular graph convolutions: Moving beyond fingerprints. *J Comput Aid Mol Des*, 2016.

[Kipf. and Welling, 2017] N. Thomas Kipf. and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[Li *et al.*, 2021a] Pengyong Li, Yuquan Li, and others. Trimnet: learning molecular representation from triplet messages for biomedicine. *BIB*, 2021.

[Li *et al.*, 2021b] Pengyong Li, Jun Wang, and others. An effective self-supervised framework for learning expressive molecular global representations to drug discovery. *BIB*, 2021.

[Liu *et al.*, 2019] Shengchao Liu, Mehmet Demirel, and others. N-gram graph - simple unsupervised representation for graphs, with applications to molecules. *NeurIPS*, 2019.

[Lu *et al.*, 2019] Chengqiang Lu, Qi Liu, and others. Molecular property prediction: A multilevel quantum interactions modeling perspective. *AAAI*, 2019.

[Ma *et al.*, 2017] Xuezhe Ma, Yingkai Gao, and others. Dropout with expectation-linear regularization. 2017.

[Mannhold *et al.*, 2012] Raimund Mannhold, Hugo Kubinyi, and others. *Bioisosteres in medicinal chemistry*. 2012.

[Mohri *et al.*, 2012] Mehryar Mohri, Afshin Rostamizadeh, and others. Foundations of machine learning. 2012.

[Ramsundar and others., 2015] B. Ramsundar and others. Massively multitask networks for drug discovery. 2015.

[Rogers and Hahn, 2010] David Rogers and Mathew. Hahn. Extended-connectivity fingerprints. *J chem inf*, 2010.

[Rong *et al.*, 2020] Yu Rong, Yatao Bian, and others. Self-supervised graph transformer on large-scale molecular data. *NIPS*, 2020.

[Schuett *et al.*, 2017] T. K. Schuett, Kindermans, and others. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *NIPS*, 2017.

[Song *et al.*, 2020] Ying Song, Shuangjia Zheng, and others. Communicative representation learning on attributed molecular graphs. In *IJCAI*, 2020.

[Sun *et al.*, 2021] Mengying Sun, Jing Xing, and others. Mocl: Contrastive learning on molecular graphs with multi-level domain knowledge. *KDD*, 2021.

[Susheel *et al.*, 2021] Susheel, Pan Li, and others. Adversarial graph augmentation to improve graph contrastive learning. 2021.

[Wang *et al.*, 2019] Sheng Wang, Yuzhi Guo, and others. Smiles-bert - large scale unsupervised pre-training for molecular property prediction. *BCB*, 2019.

[Wu *et al.*, 2018] Zhenqin Wu, Bharath Ramsundar, and others. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 2018.

[Xia *et al.*, 2021a] Jun Xia, Haitao Lin, Yongjie Xu, Lirong Wu, Zhangyang Gao, Siyuan Li, and Stan Z. Li. Towards robust graph neural networks against label noise, 2021.

[Xia *et al.*, 2021b] Jun Xia, Lirong Wu, Jintao Chen, Ge Wang, and Stan Z Li. Debiased graph contrastive learning. *arXiv preprint arXiv:2110.02027*, 2021.

[Xia *et al.*, 2022] Jun Xia, Lirong Wu, , Jintao Chen, Bozhen Hu, and Stan Z. Li. SimGRACE: A Simple Framework for Graph Contrastive Learning without Data Augmentation. In *Proceedings of The Web Conference 2022*. Association for Computing Machinery, 2022.

[Xiong *et al.*, 2020] Zhaoping Xiong, Dingyan Wang, and others. Pushing the boundaries of molecular representation for drug discovery with graph attention mechanism. *J Med Chem*, 2020.

[Xu *et al.*, 2021a] Minghao Xu, Hang Wang, and others. Self-supervised graph-level representation learning with local and global structure. *ICML*, 2021.

[Xu *et al.*, 2021b] Runxin Xu, Fuli Luo, and others. Raise a child in large language model: Towards effective and generalizable fine-tuning. *EMNLP*, 2021.

[Yang *et al.*, 2019] Kevin Yang, Kyle Swanson, and others. Analyzing learned molecular representations for property prediction. *J CHEM INF MODEL*, 2019.

[You *et al.*, 2020] Y. You, T. Chen, and others. Graph contrastive learning with augmentations. In *NeurIPS*, 2020.

[You *et al.*, 2021] Yuning You, Tianlong Chen, and others. Graph contrastive learning automated. *ICML*, 2021.

[Zhang *et al.*, 2021] Zaixi Zhang, Qi Liu, and others. Motif-based graph self-supervised learning for molecular property prediction. *NeurIPS*, 2021.

[Zhu *et al.*, 2021a] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. An empirical study of graph contrastive learning. 2021.

[Zhu *et al.*, 2021b] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021.

[Zolna *et al.*, 2018] Konrad Zolna, Devansh Arpit, and others. Fraternal dropout. In *ICLR*, 2018.