MACHINE LEARNING ON DRUG DISCOVERY: ALGORITHMS AND APPLICATIONS

By

Mengying Sun

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2022

ABSTRACT

MACHINE LEARNING ON DRUG DISCOVERY: ALGORITHMS AND APPLICATIONS

By

Mengying Sun

Drug development is an expensive and time-consuming process where thousands of chemical compounds are being tested and experiments being conducted in order to find out drugs that are safe and effective. Modern drug development aims to speed up the intermediate steps and reduce cost by leveraging machine learning techniques, typically at drug discovery and preclinical research stages. Better identification of promising candidates can significantly reduce the load of later processes, e.g., clinical trials, saving tons of resources as well as time.

In this dissertation, we explored and proposed novel machine learning algorithms for drug discovery from the aspects of robustness, knowledge transfer, molecular generation and optimization. First of all, labels from high-throughput experiments (e.g., biological profiling and chemical screening) often contain inevitable noise due to technical and biological variations. We proposed a method (RCL) that leverages both disagreement and agreement among deep neural networks to mitigate the negative effect brought by noisy labels and better predict drug responses. Secondly, graph neural networks (GNNs) has become popular for modeling graph-structured data (e.g., molecules). Graph contrastive learning, by maximizing the mutual information between paired graph augmentations, has been shown to be an effective strategy for pretraining GNNs. However, the existing graph contrastive learning methods have intrinsic limitations when adopted for molecular tasks. Therefore we proposed a method (MoCL) that utilizes domain knowledge at both local- and global-level to assist representation learning. The local-level domain knowledge guides the augmentation process such that variation is introduced without changing graph semantics. The global-level knowledge encodes the similarity information between graphs in the entire dataset and helps to learn representations with richer semantics. Last but not least, we proposed a search-based approach (MolSearch) for multi-objective molecular generation and optimization. We show that given proper design and suf-

ficient information, search-based methods can achieve performance comparable or even better than deep learning methods while being computationally efficient. Specifically, the proposed method starts with existing molecules and uses a two-stage search strategy to gradually modify them into new ones, based on transformation rules derived from large compound libraries. We demonstrate all the proposed methods with extensive experiments.

To sum up, RCL enables accurate prediction of drug-induced gene expression change which lays the foundation of virtual drug screening based on reversing signatures of a given disease. Any methods that utilizing graph neural networks for molecular tasks can utilize MoCL to improve prediciton accuracy given insufficient data. MolSearch enables generation of molecules with desired properties as well as optimizing targeted properties for better drug candidates.

This thesis is dedicated to my parents, Xiaojuan Dai and Jianjiang Sun,
as well as my husband Dr. Deliang Yang.

ACKNOWLEDGEMENTS

I would not have the opportunity to accomplish this thesis without the guidance from my dissertation committee, the help from my colleagues and friends, as well as the support from my family. I owe my sincere gratitude to all of them.

First and foremost, I would like to thank my advisor Dr. Jiayu Zhou, who gave inspiration, technical advice, and emotional support throughout my Ph.D. study. He advised me from all the aspects including conducting research, writing papers, and presenting my work to others. These experiences are lifelong treasures to me. I also would like to thank my co-advisor Dr. Bin Chen, who provided delicate expertise in system biology and drug discovery to guide my research direction. His endless passion and persistent patience for doing good research always impacted me. I would like to thank Dr. Fei Wang, who introduced me to computational drug discovery during my research internship at Weill Cornell Medicine. I would like to thank Dr. Gustavo de los Campos, who brought me into the field of quantitative genetics which lays a good foundation for the later research. I would like to thank Dr. Zhangyang Wang, Dr. Liang Zhan, and Dr. Jinfeng Yi whom I collaborated with during my early Ph.D. study. I would like to thank Dr. Pang-Ning Tan and Dr. Kevin J. Liu to serve as my committee members and help me to accomplish this dissertation.

Moreover, I would like to thank my closest collaborator, Dr. Jing Xing, who greatly supported all my projects. Jing is a professional and passionate researcher in computational drug discovery and system biology. We worked together through late nights and deadlines during the past 4 years. I learned from and was impacted so much by her, from professional knowledge to life philosophy. I am also thankful to my great colleagues Dr. Inci M. Baytas, Dr. Qi Wang, Fengyi (Andy) Tang, and Boyang Liu whom I have worked together with in multiple projects. I am thankful to my ILLIDAN lab-mates Dr. Kaixiang Lin, Liyang Xie, Junyuan Hong, Zhuangdi Zhu, Ikechukwu Uchendu, Shuyang Yu for a productive, friendly, and supportive academic environment. I would like to thank all members from Bin Chen Lab at MSU and from Dr. Fei Wang's lab at Weill Cornell Medicine for providing a great environment during my stay. I thank all members from QuantGen Group at

MSU during my master's study for building solid foundations on quantitative genetics.

I want to express my sincere thanks to Dr. Huiyun Wu, who has been a friend to me since I came to the United States. We help and support each other closely during the past eight years and I cannot express how grateful I am to meet her in my life. I would also like to thank Jing Shang, Siyi Hou, and Dr. Chongguang Bi who support me during my early time in the United States. To all my friends, thank you all for your support and I wish you the best of luck wherever you are.

My sincere gratitude also goes to my mother, Xiaojuan Dai, and my father, Jianjiang Sun. Their unconditional love, sacrifice, support, and encouragement accompany me in every step of my life. It's their hardworking that enables me to pursue my own life freely and securely. I would not be me without them.

Finally, I would like to thank my soulmate Dr. Deliang Yang, with whom I spent the most amazing time in my life. We stood by each other and walked through all the good and bad days during the past five years. We enjoy every moment in our current life and share the same goals towards an exciting future. My life just becomes much more meaningful as I met him, and our two cats Taiji and Huita.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

1.1 Robust Learning on Noisy Labels

Recent years have witnessed huge successes of supervised learning using deep neural networks in various domains [39, 101, 29]. Moreover, investigations on network property and behavior have further brought a better understanding of deep models, which in turn provides guidance on their utilization [4, 92]. One decisive factor behind such successes is the availability of a sufficiently large amount of training data [108].

In fact, improving generalization from limited labeled data has been an active research topic for an extended period of time. For example, semi-supervised learning aims to help learning by leveraging the unlabeled data [131, 16, 15]. On the other hand, in cases where obtaining accurate labels is too expensive, practitioners could use affordable and alternative apparatuses to collect less reliable *noisy* labels. For example, the online crowd-sourcing tools such as Mechanical Turk [1], in which although the labels come from humans, the quality of labels varies among different annotators, and even within the same annotator across different time. Besides, labels from high-throughput experiments (e.g., biological profiling and chemical screening) often contain inevitable noise due to technical and biological variations.

Learning with noisy labels has imposed additional challenges. Sometimes the data quality is known *a priori* [65, 97, 27], but a more common scenario is that the data available is a mixture of samples with both clean and noisy labels and one does not know, or only has partial knowledge of the underlying distribution of the noise [76, 66, 74, 118]. In this problem setting, a learning process that is aware of noise in the labels and actively mitigates the negative impacts from the noisy labels, is the key to improving the generalization of learned models.

Among the early studies, [12] showed that a proper arrangement of tasks can improve con-

[1]https://www.mturk.com/

vergence and generalization when training a deep neural network. They proposed the concept of *curriculum*, which defines the order of learning tasks, usually from easy to hard, to assist network training. Recently, the memorization effect of neural networks has been identified and analyzed in [4], showing that neural networks tend to fit informative information first, such as simple patterns, and then the non-informative part, such as noise. Therefore, an appropriate ordering of data may also improve the robustness of deep models. On the other hand, instead of manually grouping data, [59] introduced a latent variable associated with each sample as the curriculum (sample weight) and learns simultaneously with the model parameter. Several pre-defined curriculums have been proposed later in [111, 64, 48, 49, 138] and such learning process is referred to as self-paced learning (SPL).

Though slightly deviate from the initial motivation, learning with curriculums reveals its power especially in dealing with noisy data. The reason is that noisy samples can be re-weighted or even filtered out via the curriculum mechanism under proper designs. It has been proved by [73] that the latent objective of self-paced learning is equivalent to a robust loss function, which also sheds lights on the effectiveness of SPL on noisy data. The original optimization of SPL can be done by alternatively optimizing model parameters and the curriculum, known as the majorization-minimization (MM) algorithm [62]. However, such procedure is intractable for training very large and deep neural networks via mini-batch stochastic gradients. Therefore, [50] modified the algorithm such that it optimizes both the model parameter and curriculum stochastically over mini-batches in a more elegant way. The authors also proposed to learn the curriculum rather than pre-define and optimize it when auxiliary data is available.

A major drawback of determining curriculum based on the learner's own ability without any other supervision or feedback is the sample selection bias from the learner itself. The error made in the early stage will be enhanced as training proceeds. Therefore, two or more networks have been introduced in recent works to mitigate the selection bias [40, 63]. Nevertheless, these studies either emphasize the disagreement or focus on the agreement between networks only without considering the other.

1.1.1 Our Contribution

In this paper, we first investigate the mechanism of how disagreement and agreement can help filter out noisy samples. Then based on the insights, we propose a novel framework called **R**obust **C**ollaborative **L**earning (**RCL**) to deal with noisy labels. The main contributions of this paper are:

- We show that *disagreement* between networks can diversify the gradients of model weights from noisy samples, which slows down the accumulation of noisy gradients.
- We show that under certain conditions, *agreement* from more than one network can improve the quality of data selection, i.e., the label purity increases.
- Combining the above two findings, we propose RCL framework that consists of multiple networks, where each network is an individual learner and exchanges knowledge with its *Peer* system. The knowledge of the *Peer* system is fused from multiple networks, by adaptively encouraging *disagreement* in the early stage and *agreement* in the later stage, which fully boost the selection of clean samples for training.

We demonstrate the effectiveness of RCL on both synthetic and real data experiments. For synthetic experiment, we use the benchmark image data [57] under different noise settings following literature [40, 117]. We further validated our framework on cancer drug development using large-scale genomic data from multiple sources [105, 124]. The proposed method achieves state-of-art performance and significantly outperforms baselines in large noise settings, on both image and bioinformatics data.

1.2 Graph Neural Networks and Pretraining Strategies

Graph neural networks (GNNs) has been demonstrated to achieve state-of-the-art performance on graph-related tasks such as node classification [54, 119, 126], link prediction [142] and graph classification [119, 35, 129]. It has also been frequently used in the biomedical domain recently to tackle drug-related problems [104, 94, 75]. However, like most deep learning architectures, it requires a large amount of labeled data to train whereas task-specific labels in the real world are often of limited size (e.g., in the biomedical domain, requiring labels such as drug responses from biological experiments is always expensive and time-consuming). Therefore, pretraining schemes on GNNs have been actively explored recently.

One line of works focuses on designing pretext tasks to learn node or graph representations without labels. The predefined tasks include graph reconstruction [55, 44, 135] and context prediction [83, 43]. The other line follows contrastive learning framework from the computer vision domain [21, 127], in which two augmentations are generated for each data and then fed into an encoder and a projection head. By maximizing the mutual information between the two augmented views, the model is able to learn representations that are invariant to transformations. In particular, [134] proposed four types of augmentations for general graphs and demonstrated that contrastive learning on graphs can produce representations that are beneficial for downstream tasks.

However, unlike images, contrastive learning on graphs has its unique challenges. First, the structural information and semantics of the graphs vary significantly across domains (e.g., social network v.s. molecular graphs), thus it is difficult to design a universal augmentation scheme that fits all scenarios. It has been shown that general augmentations can be harmful under a specific domain context [134]. Second, most current graph contrastive learning frameworks learn invariant representations while neglect the global structure of the entire data [5], e.g., some graphs should be closer in the embedding space due to their structural similarity. Nevertheless, modeling similarity between graphs itself is still a difficult problem [8]. Third, the contrast schemes are not unique because graph tasks can happen at different levels, e.g., node-graph contrast [42], node-node contrast [141], graph-graph contrast [134] are all possible contrast schemes.

Besides these unique challenges for graphs, contrastive learning itself also has unsolved problems. For example, accurately estimating mutual information in high dimension is difficult [84]. The connection between mutual information maximization and the success of contrastive learning is still not clear. In fact, [114] found the connection is actually weak, while instead metric learning shares some intrinsic connections with contrastive learning. These findings also motivate us to pay more attention to the role of augmentation schemes and global semantics of the data in order to improve contrastive learning on graphs.

1.2.1 Our Contribution

In this paper, we aim to tackle the aforementioned challenges in the context of biomedical domain, where molecular graphs are present. Our hypothesis is that better representations can be learned by infusing domain knowledge into the augmentation and contrast schemes. We propose to leverage both local-level and global-level domain knowledge to assist contrastive learning on molecular graphs. In particular, unlike general augmentations in which nodes and edges in a graph are randomly perturbed, we propose a new augmentation scheme called substructure substitution such that a valid substructure in a molecule is replaced by a bioisostere which introduces variation without altering the molecular properties too much. The substitution rules are derived from domain resources and we regard it as local-level domain knowledge. The global-level domain knowledge encodes the global similarities between graphs. We proposed to utilize such information to learn richer representations via a double-contrast objective.

Leveraging domain knowledge to assist contrastive learning has rarely been explored in literature and our work is the first to make this attempt. In summary, our contributions are as follows:

- We propose a new augmentation scheme for molecular graphs based on local-level domain knowledge such that the semantics of graphs do not change in the augmentation process.
- We propose to encode the global structure of data into graph representations by adding a global contrast loss utilizing the similarity information between molecular graphs.
- We provide theoretical justifications that the learning objective is connected with triplet loss in metric learning which shed light on the effectiveness of the entire framework.
- We evaluate MoCL on various molecular datasets under both linear and semi-supervised settings and demonstrate its superiority over the state-of-the-art methods.

1.3 Search-based Molecular Generation and Property Optimization

Searching new compounds with desired properties is a routine task in early-stage drug discovery [14]. Common examples include improving the binding activity against one or multiple therapeutic targets while keeping the drug-likeness property; increasing drug solubility while minimizing the change of ADME properties. However, a small change of chemical structures may lead to an unwanted challenge of one property that even seasoned chemists cannot foresee. Moreover, the virtually infinite chemical space and the diverse properties for consideration impose significant challenges in practice [96]. Advanced machine learning models built upon historical biological and medicinal chemistry data are poised to aid medicinal chemists in designing compounds with multiple objectives efficiently and effectively.

Leveraging computational methods to facilitate and speed up the drug discovery process has always been an active research area [102, 136]. In particular, using deep learning (DL) and reinforcement learning (RL) to generate and optimize molecules has recently received broad attentions [51, 133, 128], which we will summarize in detail later in section **??**. Despite the advances, such methods either rely on the quality of latent space obtained by generative models [98], or suffer from high variation, making it hard to train [112]. In reality, DL/RL methods consume large computational resources while the generated molecules hardly synthesize. Methods combing multiple objectives often do not work well [31].

In this paper, instead of leveraging DL, we propose a practical search-driven approach based on Monte Carlo tree search (MCTS) to generate molecules. We show that under proper design, search methods can achieve comparable or even better results to DL methods in terms of multi-objective molecular generation and optimization, while being computationally much more efficient. *The efficiency and multi-objective nature allow it to be readily deployed in massive real-world applications such as early-stage drug discovery.*

In order to design an efficient and effective search framework for practical multi-objective molecular generation and optimization, we need to answer the following questions. *Q1*: where to start; *Q2*: what to search; and *Q3*: how to search. For Q1, prior works that use MCTS to generate molecules mostly start with empty molecules [130, 46]. Since most drug-like molecules have 10-40 atoms, the search tree can grow very deep and the search space grows exponentially with the depth, which makes the search process less efficient and effective. Some work thus uses pre-trained RNN as a simulator to expand the tree however it requires additional pretraining [130]. Moreover, real optimization projects often have some candidates in place. For Q2, most prior works use atom-wise actions for editing molecules, which makes it hard to improve target property while maintaining drug-likeness and synthesis abilities [140, 133]. Fragment-wise actions tend to work better but the editing rules are mostly heuristic [52, 128]. For Q3, most existing methods combine all the objectives into one single score and optimize for that [78, 128]. However, the simple aggregation of scores neither fully considers the differences of objective classes nor reflects real optimization scenario.

We seek solutions to Q1-Q3 and propose MolSearch, a simple and practicable search framework for multi-objective molecular generation and optimization. In MolSearch, we start with existing molecules and optimize them towards desired ones (Q1). The modification is based on design moves [7], i.e., transformation rules that are chemically reasonable and derived from large compound libraries (Q2). The property objectives are split into two groups with its rationale explained in detail later. The first group contains all biological properties such as inhibition scores to proteins, and the second group includes non-biological properties such as drug-likeness (QED) and synthetic accessibility (SA). Correspondingly, the entire search process consists of two stages: a HIT-MCTS stage that aims to improve biological properties, followed by a LEAD-MCTS stage that focuses on non-biological properties while keeping biological ones above certain threshold. Each stage contains a multi-objective Monte Carlo search tree where different property objectives are considered separately rather than combined (Q3).

We evaluate MolSearch on benchmark tasks under different generation settings and compare it with various baselines. The results show that MolSearch is on par with or even better than the baselines based on evaluation metrics calculated from success rate, novelty and diversity, within much less running time.

1.3.1 Our Contribution

- MolSearch is among the first that make search-based approaches comparable to DL-based methods in terms of multi-objective molecular generation and optimization.
- MolSearch combines mature components, e.g., tree search, design moves, multi-objective optimization, in a novel way such that the generated molecules not only have desired properties but also achieve a wide range of diversity.
- MolSearch is computationally very efficient and can be easily adopted into any real drug discovery projects without additional knowledge beyond property targets.
- Additional to molecular generation, MolSearch is more tailored for hit-to-lead optimization given the nature of its design, which makes it very general and applicable.

CHAPTER 2

RELATED WORK

2.1 Robust Learning on Noisy Labels

Inspired by the fact that humans learn better when trained with a curriculum-like strategy, [12] first proposed curriculum learning, which mimics the learning procedure of humans. Results on both visual and language tasks have shown that training on easy task first and then hard tasks led to faster convergence as well as better generalization. Instead of using a specified curriculum, [59] incorporated a latent variable associated with each sample, and jointly optimized the model parameters and the sample curriculum. Besides, a variety of approaches with different predefined curriculum were proposed and validated [111, 64, 48, 49, 138]. Further, instead of using predefined curriculum, [50] proposed to learn a data-driven curriculum when auxiliary data is available. The authors also designed an efficient algorithm for training very deep neural networks with curriculum. Later, [40] proposed co-teaching framework, a system of two networks that exchange selected samples to alleviate the sample selection bias brought by one network. Co-teaching [40] works well empirically with several follow-up works and applications [137, 122, 103]. [103] later proposed to make use of the unselected samples by correcting their labels and combining them with selected samples for training. Other very recent works also aggregated knowledge from multiple sources, e.g., multiple networks or multiple training epochs of a single network to filter out noisy data [63, 77].

Learning with corrupted labels also relates to weak supervised learning, and its recent advances can be summarized into the following groups. A common way to leverage weak labels when the quality of data is known *a priori* is to use the pre-train and fine-tune scheme based on the amount of clean and weak data [28, 97, 65]. Another line of methods design surrogate loss functions for robust learning [70, 66, 76]. Some approaches model the noise pattern or estimate the error transition matrix [107, 74], and denoise by either adding an extra layer [36], or using generative models [88, 118]. Other methods utilize semi-supervised learning techniques [1, 24] to revise weak labels for further training [41], or regularize the learning procedure [115]. Recently, learning-to-learn methods have also been proposed to tackle such problems by manipulating gradient update rules [3, 27]. Since our work mainly follows curriculum learning, we do not expose further details for works mentioned in this paragraph and refer readers of interest to the original papers.

2.2 Graph Neural Networks and Pretraining

Self-supervised learning on graphs. A common strategy for learning node (graph) representation in an unsupervised manner is to design pretext task on unlabled data. For node-level tasks, You et al. [135] proposed three types of self-supervised tasks: node clustering, graph partition and graph completion to learn node representations. Peng et al. [83] proposed to predict the contextual position of a node relative to the other to encode the global topology into node representations. GPT-GNN [44] designed generative task in which node attributes and edges are alternatively generated such that the likelihood of a graph is maximized. After that, the pretrained GNN can be used for any down-stream tasks. For graph level tasks, Hu et al. [43] first designed two tasks, predicting neighborhood context and node attributes to learn meaningful node representations, then using graph-level multi-task pretraining to refine the graph representation. Other works [99, 132, 110] utilized similar strategies for either node or graph level pretrain in the context of a more specific task or domain.

Contrastive learning on graphs. Contrastive learning on graphs can be categorized into two groups. One group aims to encode structure information by contrasting local and global representations. For example, DGI [120] proposed to maximize the mutual information between node embedding and graph summary vector to learn node representations that capture the graph semantics. InfoGraph [109] extended DGI to learn graph-level representations and further proposed a variant for semi-supervised scenarios. Another group aims to learn representations that are invariant to transformations, following the idea of contrastive learning on visual representations [21, 127, 30], where two augmentations (views) of an image are generated and fed into an encoder and a projection head, after which their mutual information is maximized. Similarly, You et al. [134] explored four types of augmentations for general graphs and demonstrated that the learned representations can help down-streaming tasks. Instead of general corruption, [42] used graph diffusion to generate the second view and performed contrast between node and graph from two views. GCC [85] proposed to use random walk to generate subgraphs and contrast between them. GCA [141] proposed adaptive augmentation such that only unimportant nodes and edges are perturbed. However, GCA is focused on network data and not suitable for molecular graphs.

Evaluation protocols. Since our work focus on graph-level representation learning, and there are various evaluation schemes for graph self-supervised learning, we summarize the evaluation protocols that related works use. Most prior works [109, 43, 134] adopt the linear evaluation protocol where a linear classifier is trained on top of the representations. [109, 134] also adopt the semi-supervised protocol where only a small fraction of labels are available for downstream tasks. [43, 134] explore the transfer learning setting in which the pretrained model is applied to other datasets.

2.3 Molecular Generation and Optimization

In general, molecular property optimization comprises three components or less: representation, generative model, and optimization model. The representation of molecules can be simplified molecular-input line-entry system (SMILES) strings, circular fingerprints, and raw graphs, which often corresponds to certain type of generative models. Grouping by each component can be too detailed to capture the big picture, therefore we choose to categorize the related studies based on optimization models.

The first group optimizes molecular via Bayesian optimization [37, 60, 25, 51]. These methods first learn a latent space of molecules via generative models such as auto-encoders (AEs), then optimize the property by navigating in that latent space, and generates molecules through the decoding process. Most methods in this category only optimize for non-biological properties such as QED and penalized logP [1], and focus on metrics such as validity of generated molecules. They heavily rely on the quality of learned latent spaces, which impose challenges for multi-objective optimization.

Instead of manipulating latent representations, the second category utilizes reinforcement learning (RL) to optimize molecular property. One line of research applies policy gradient to finetune generative models, e.g., GAN-based generator [95, 26], GNN-based generator [133], Flow-based generator [100, 67] to generate molecules with better property scores. The other line of work directly learns the value function of molecule states and optimizes for a given property via double Q-learning [140].

Besides RL, the third category uses genetic algorithms (GAs) to generate molecules with desired properties [46, 2, 78]. The generation process of genetic algorithms usually follows mutation and cross-over rules that are predefined from a reference compound library or domain expertise, which are not easy to obtain in general. Some work [78] also combines deep learning, e.g., a discriminator into GA generator to increase the diversity of molecules.

The last but least explored category aims to optimize molecular property using search methods, e.g., Monte Carlo tree search (MCTS). The earliest work traces back to [130, 46] in which the authors uses pre-trained RNNs or genetic mutation rules as the simulator for tree expansion and simulation. [86] proposes atom-based MCTS method without predefined simulator. Again, all the methods focus on single and non-biological properties and are not tailored for multi-objective optimization. Not until recently RationaleRL [52] enables multi-objective molecular generation by first searching property-related fragments using MCTS and then completing the molecular graph using reinforcement learning.

There are also pioneering works that do not fall into any of the categories above, e.g., MARS [128] proposes a Markov sampling process based on molecular fragments and graph neural networks (GNNs) and achieves state-of-the-art performance. In summary, we see a trend of utilizing fragment-based actions and directly navigating in the chemical space (a.o.t. generative models) in

[1] water-octanol partition coefficient penalized by synthesis accessibility and number of cycles having more than 6 atoms, i.e., PlogP(m)=logP(m)-SA(m)-cycle(m)

recent works. Interested readers are referred to [31, 139] for a comprehensive understanding of advances in molecular generation and optimization.

CHAPTER 3

ROBUST COLLABORATIVE LEARNING WITH NOISY LABELS

3.1 A Revisit of Self-paced Learning (SPL)

Despite the promising evidence of curriculum in assisting learning, constructing an effective curriculum is not easy for learning tasks. To tackle this challenge, SPL [59] introduces a latent variable associated with each training sample, and solve them during training. After denoting the latent variables in a vector $\mathbf{v} \in [0,1]^n$, where n is the sample size, the objective function of SPL can be written as:

$$\min_{\mathbf{w},\mathbf{v}\in[0,1]^n} \mathbf{E}(\mathbf{w},\mathbf{v},\lambda) = \sum\nolimits_{i=1}^{n} v_i L\big(y_i, f(\mathbf{x}_i,\mathbf{w})\big) + g(v_i,\lambda) \tag{3.1}$$

where $g(v,\lambda)$ serves as the curriculum function and regularizes the weight of a given sample, λ is a control parameter. $\mathbf{w}$ and $\mathbf{v}$ are optimized alternatively while fixing the other [116]. A simple example of the curriculum function can be $g(v,\lambda) = -\lambda v$ with closed-form solution for v at each step:

$$v^*(\lambda; l) = \begin{cases} 1, & \text{if } l < \lambda \\ 0, & \text{if } l \geq \lambda \end{cases}$$

where l is the loss for one sample. The design of $g(v,\lambda)$ often satisfies conditions such as convexity and monotonicity to ensure convergence [47, 49, 73]. Moreover, such design reveals the nature of SPL, that the difficulty of a sample is determined by the learner's ability, i.e., if a sample has large loss on the current model, it is likely to be more difficult to learn or even an outlier. It has been proved that the optimization of Eq. (3.1) is equivalent to minimizing a robust loss function, whose underlying objective is $F_\lambda(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\int_0^{l_i} v^*(\lambda,l)dl$ [73]. Therefore, SPL works well for problems involving data with corrupted labels. Furthermore, the alternating minimization algorithm for solving Eq. (3.1) can be modified to suit the mini-batch training for very deep neural networks [50].

3.1.1 The power of Disagreement

A major drawback of SPL is the sample selection bias induced by the learner itself since it picks samples based on its own knowledge. The error that takes place in the early stage will be reinforced as training continues. In order to mitigate this, a second network is introduced in co-teaching [40], where two networks first pick samples on their own and then exchange selected samples to train. Such strategy works better than SPL in practice. However, the underlying mechanism has not been well studied. Therefore, in this subsection, we show that exchanging data introduces disagreement between two networks and such disagreement can diversify noisy gradients and thus leads to higher gradient purity.

Recall that the algorithms of many machine learning models are based on gradient methods, in which the model is updated iteratively by adding gradients computed from the samples. Therefore the final learned model is additive w.r.t. the gradients. In the stochastic gradient procedure, the gradient update for each mini-batch can be written as:

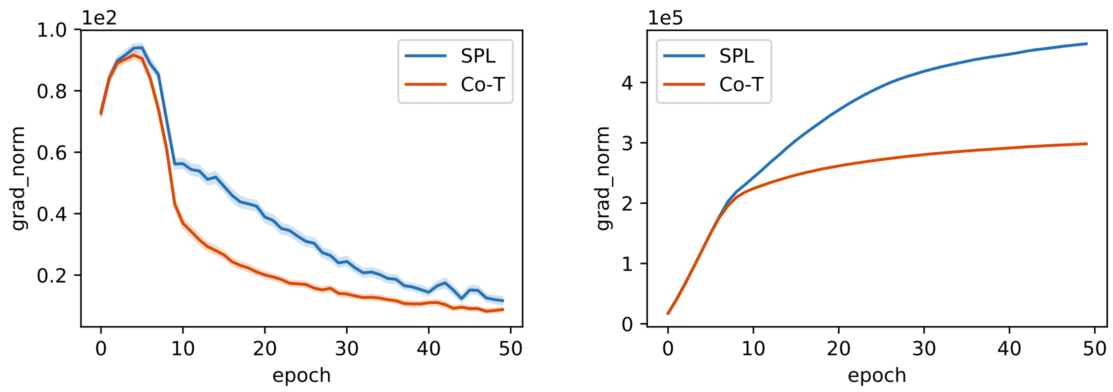
$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta^t \frac{1}{n_r} \sum\nolimits_{j=1}^{n_r} \nabla l_j(\mathbf{w}^t) = \mathbf{w}^t - \eta^t \frac{1}{n_r} S_\nabla,$$

where n_r is the number of samples in a mini-batch, η^t is the step size, S_∇ denotes the summation of sample gradients.

Considering each of the two networks, with an oracle that provides the ground truth whether a label is noisy or not, we can decompose the gradient S_∇ into four disjoint components based on data quality (clean or noisy) and network agreement (agree or not). Let I be a subset of indexes such that 11={clean, agree}, 10={clean, disagree}, 01={noisy, agree}, 00={noisy, disagree}, then the decomposition is given by:

$$S_\nabla = \underbrace{\Sigma_{j\in I_{11}} \nabla l_j(\mathbf{w}^t)}_{\text{agreed clean}} + \underbrace{\Sigma_{j\in I_{10}} \nabla l_j(\mathbf{w}^t)}_{\text{disagreed clean}} + \underbrace{\Sigma_{j\in I_{01}} \nabla l_j(\mathbf{w}^t)}_{\text{agreed noisy}} + \underbrace{\Sigma_{j\in I_{00}} \nabla l_j(\mathbf{w}^t)}_{\text{disagreed noisy}},$$

where an ideal algorithm should concentrate the learning clean data points $I_{11} \cup I_{10}$ and reduce the impacts from noisy data points $I_{01} \cup I_{00}$. As SPL updates network parameters based on small loss samples from itself, the network goes towards the small loss direction, thus the similarity of noisy



(a) GN at each epoch (b) Accumulative GN

Figure 3.1: Gradient norm (GN) of *noisy* data on CIFAR10. The gradient includes components I_{01} and I_{00} in Eq. (3.2). Co-teaching learning process has less impact from noisy data.

samples will concentrate and accumulate. In co-teaching, on the other hand, the set I_{00} (noisy, disagree) are diversified by exchanging data points between the two networks. Such disagreement is crucial and can cause several effects. First, the gradient norm of noisy data may diminish; second, the diversification can take effect across time since gradients are eventually summed and applied to network parameters; third, introducing disagreement is equivalent to adding small perturbations on network parameters, which could increase the robustness of the network. We note that the noise from I_{01} (noisy, agree) set of samples is inevitable since they are agreed by both two networks, but such "bad" agreements may also suggest the usefulness of the samples and effectively prevent overfitting.

We evaluate these effects in a small synthetic experiment. Given an image-classification problem, e.g., CIFAR10, for each class, we manually flip 45% labels into the adjacent class. Then we compare the gradients of noisy samples between SPL and co-teaching, i.e., $\Sigma_{j\in I_{01}}\nabla l_j(\mathbf{w}^t) + \Sigma_{j\in I_{00}}\nabla l_j(\mathbf{w}^t)$ in Eq. (3.2). The gradients are calculated from the last linear layer of a CNN model. We use gradient norm here since gradient itself cannot be directly compared. Fig. 3.1a shows the average gradient norm of noisy data at each epoch, and Fig. 3.1b shows the norm of accumulative gradients of noisy data along time. We can see that disagreement from exchanging data helps achieve smaller noisy gradient compared to SPL and it also slows down the accumulation of noisy gradients.

3.1.2 The power of Agreement

The training procedure is a dynamic process in which the leaner's ability grows as training proceeds. When the learners are mature, aggregating their knowledge can be beneficial as compared to only exchanging them.

Lemma 1 *Given two networks, the samples selected by either network can be decomposed into two subsets: agreement ($A = I_{11} \cup I_{01}$) and disagreement ($\bar{A} = I_{01} \cup I_{00}$) samples as compared to the other. Define $p_I = \frac{n_c}{n_c + n_{\bar{c}}}$ as the purity of a subset I, $n_c = |I_{11}|, n_{\bar{c}} = |I_{10}|$. If $p_A > p_{\bar{A}}$, then $p_A > p_{A \cup \bar{A}}$.*

$$\begin{aligned}
p_A > p_{\bar{A}} &\Leftrightarrow \frac{n_{11}}{n_{11}+n_{01}} > \frac{n_{10}}{n_{10}+n_{00}} && \text{(by definition)} \\
&\Leftrightarrow n_{11}n_{00} > n_{10}n_{01} && \text{(simplify)} \\
&\Leftrightarrow n_{11}(n_{00}+n_{01}) > (n_{10}+n_{11})n_{01} && \text{(add } n_{11}n_{01}\text{)} \\
&\Leftrightarrow \frac{n_{11}}{n_{01}} > \frac{n_{10}+n_{11}}{n_{00}+n_{01}} = \frac{n_{1\cdot}}{n_{0\cdot}} && \text{(re-arrange)} \\
&\Leftrightarrow \frac{1}{1+\frac{n_{01}}{n_{11}}} > \frac{1}{1+\frac{n_{0\cdot}}{n_{1\cdot}}} && \text{(reciprocal twice)} \\
&\Leftrightarrow \frac{n_{11}}{n_{11}+n_{01}} > \frac{n_{1\cdot}}{n_{1\cdot}+n_{0\cdot}} && \text{(simplify)} \\
&\Leftrightarrow p_A > p_{A \cup \bar{A}} && \text{(by definition)}
\end{aligned}$$

Lemma 1 implies that for two networks, when the purity of agreed samples is larger than that of disagreed samples, the common samples selected by two networks is guaranteed to have higher purity than those selected by a single network. The proof is straightforward using definition.

In fact, recent works [63, 77] propose to aggregate knowledge from multiple networks to filter out noisy samples during training and show promising results. However, ensemble does not guarantee higher purity especially in the early training stage since errors could also be magnified.

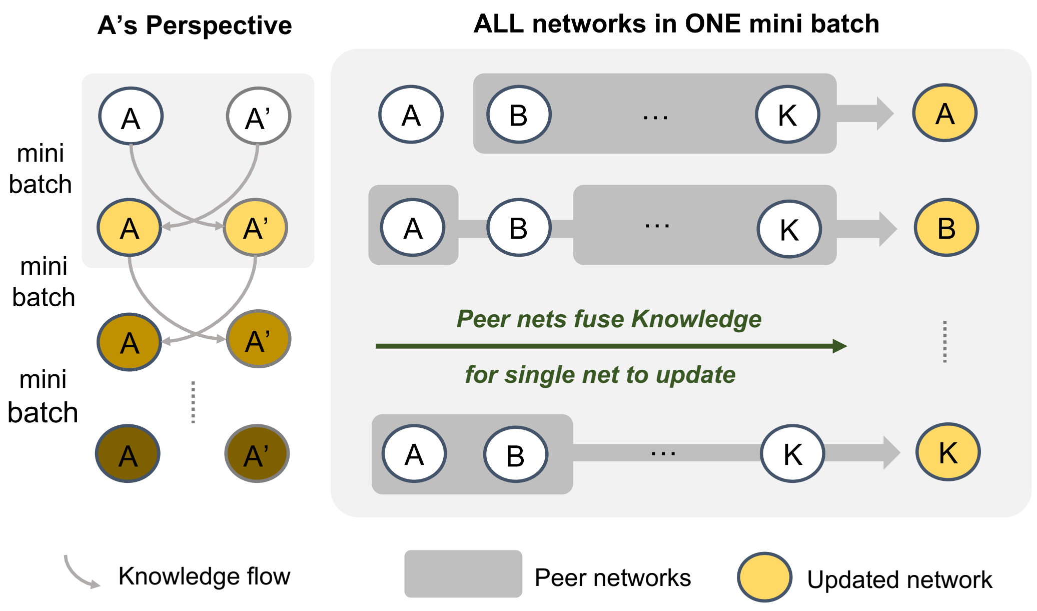


Figure 3.2: Robust Collaborative Learning (RCL) framework. Left: one network's view, A' denotes all the peer networks of A; Right: all networks in one mini batch. Knowledge fusion and update can be done in parallel for all the networks.

Therefore, a common strategy is to train the entire data until certain epochs and then perform the ensemble. The performance of such ensemble methods depends heavily on the warm-up procedure in which the entire data is used to train, therefore, if the noise rate is large, these methods may not be optimal.

3.2 The Proposed Method

The analysis of disagreement and agreement in previous section shows their advantages but also reveals the fact that focusing on either one alone while ignoring the other may lead to suboptimal results. Therefore, in this paper, we propose Robust Collaborative Learning (RCL), a framework that combines the aforementioned ingredients in a coherent way. Fig. 3.2 shows the overall structure. In RCL, each network is an individual learner, while the rest networks form a *Peer* system. From each network's perspective, it exchanges knowledge with its *Peer* (Fig. 3.2 Left). The knowledge it receives is fused from multiple networks in the *Peer* system, while the knowledge it offers will wait for fusion when itself is served as a peer network (Fig. 3.2 Right). The pseudo code of the overall algorithm is illustrated in Alg. 5.1. Next we introduce each component of RCL in detail.

3.2.1 Self-Knowledge

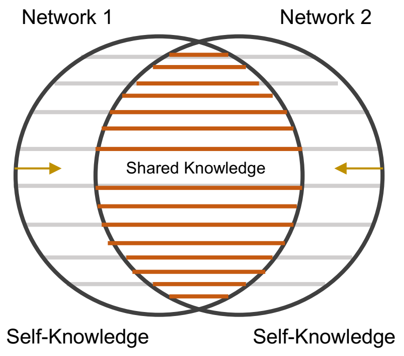
During one mini-batch, each network first selects reliable samples on its own. The selection is based on current loss such that top $R \times 100\%$ ranked small-loss samples will be selected. Due to the memorization effect in deep models [4], i.e., deep neural networks tend to learn easy patterns first and then memorize noise at later epochs, the reserve rate $R(T)$ is designed to be monotonically decreasing with respect to epoch T from 100% until it reaches clean rate $(1-\epsilon) \times 100\%$, where ϵ is the noise rate. T_{cut} is the switch epoch such that after this epoch, only $(1-\epsilon) \times 100\%$ percentage of the data will be selected if we know the noise rate in priori, otherwise ϵ itself becomes a hyper-parameter to tune. The selected samples are the self-knowledge of each individual network. After each network generates its own knowledge, the knowledge will be used in knowledge fusion step.

Algorithm 3.1: Pseudo code for RCL Algorithm.

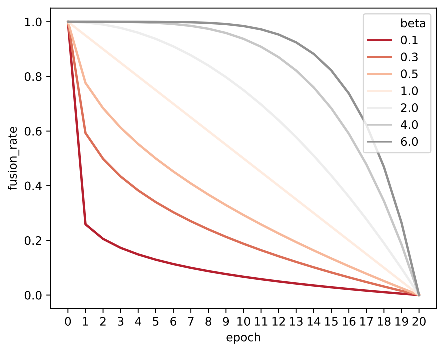
```
Input: $K$ networks $\{\Theta_1..\Theta_K\}$, training data $\mathcal{D}$, noise rate $\epsilon$; (Fixed) learning rate $\eta$, epoch $T_{\max}$
 and iteration $N_{\max}$; (Hyper) epoch $T_{\text{cut}}$, fusion multiplier $\alpha$, fusion exponent $\beta$.
Output: Updated network parameters $\{\Theta'_1..\Theta'_k\}$.
 1: for $T = 1$ to $T_{\max}$ do
 2:     Shuffle training set $\mathcal{D}$
 3:     Update $R(T) = 1 - \epsilon \cdot \min\left\{\frac{T}{T_{\text{cut}}}, 1\right\}$    // remember rate
 4:     Update $r(T) = 1 - \min\left\{\left(\frac{T}{\alpha T_{\text{cut}}}\right)^{\beta}, 1\right\}$    // fusion rate
 5:     for $N = 1$ to $N_{\max}$ do
 6:         Fetch mini-batch $D$ from $\mathcal{D}$
 7:         for $k = 1$ to $K$ do
 8:             Obtain $D_k = \arg\min_{\mathbb{D}:|\mathbb{D}|\le R(T)|D|} l(f_{\Theta_k}, \mathbb{D})$
 9:         for $k = 1$ to $K$ do
10:             Obtain $D'_k = Knowledge\left(D_{\{1..K\}\backslash k}, r(T)\right)$ Update $\Theta'_k = \Theta_k - \eta \nabla l(f_{\Theta_k}, D'_k)$
11: Return $\Theta' = \{\Theta'_1..\Theta'_k\}$;
```

3.2.2 Knowledge Fusion

For a given network k, it utilize the knowledge from its *Peer* system. The *Peer* system includes all the rest networks except for network k. The knowledge of this system is fused from multiple networks via a knowledge fusion function. Ideally, when the networks are trained well, the



(a) A 2-network example.



(b) Fusion rate w.r.t. β.

Figure 3.3: Knowledge fusion among peer networks.

knowledge of agreement, i.e., data points picked by all the peer networks can be used to update network k. However, during the early stage, the networks have not learned well and are prone to making mistakes. Therefore, disagreement is introduced to reduce the noise in gradients. There are two parameters associated with it. The first one is α, which determines the switch epoch between disagreement and agreement. Instead of setting a particular epoch, we design α as a parameter that defines the lag of switch epoch compared to T_{cut}. The second one is fusion rate r, which controls the strength of disagreement, i.e., the proportion of disagreed samples that will be included in addition to the common samples. As epoch increases, less disagreed samples will be included until only common ones are selected. Fig. 3.3a illustrates such procedure in the example of two peer networks. The decay of strength of disagreement is controlled by a hyper-parameter β and different decay patterns are shown in Fig. 3.3b. Small β corresponds to low disagreement strength and quickly transit the state from disagreement to agreement while large β encourages disagreement and slows down the transition. In summary, the fusion rate for each epoch can be calculated by the following function:

$$r(T) = \begin{cases} 1 - (T/(\alpha T_{\text{cut}}))^{\beta}, & \text{if } T < \alpha T_{\text{cut}} \\ 0, & \text{if } T \geq \alpha T_{\text{cut}} \end{cases} \tag{3.2}$$

where T is epoch, T_{cut} is the threshold when reserve rate R reaches clean rate, α is the lag parameter that controls the end point of disagreement decay compared to reserve rate. After calculating the fusion rate, disagreed samples are randomly picked and added to the agreed samples as the final knowledge of the *Peer* system. Such randomness also introduce certain level of disagreement since each network will receive different candidates to train even if the common samples are the same within each *Peer* system. The pseudo code of knowledge fusion procedure for multiple networks is illustrated in Alg. 4.2.

3.2.3 Knowledge Exchange

Network k receives the candidate samples from its *Peer* system and update parameters based on them. The same procedure can be done in parallel for all the networks. Although it seems that network k only receives knowledge in this round without giving out its own knowledge, but when other networks is updating, each of them uses knowledge from network k. This procedure is referred to as knowledge exchange. After all the networks have been updated, they enter the next iteration and repeat the processes.

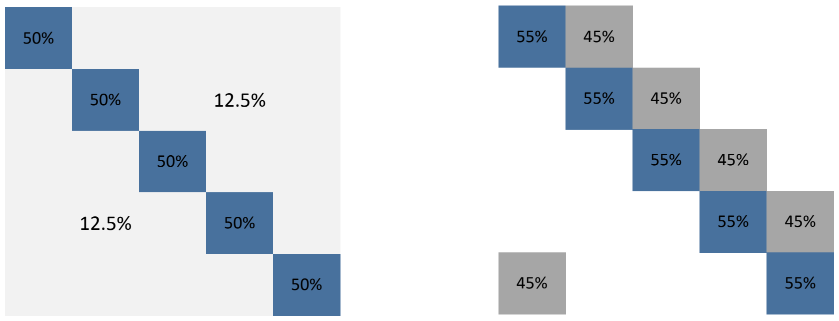
Algorithm 3.2: Knowledge Fusion Function.

```
Input: Given the k-th network, the knowledge of all other peer networks $\{D_1..D_K\} \setminus D_k$;
 Fusion rate $r(T)$.
Output: Data for updating the k-th network $D'_k$.
 1:     $D_{\text{agree}} = \text{Intersect}\left(\{D_1..D_K\} \setminus D_k\right)$
 2:     $D_{\text{potential}} = \text{Union}\left(\{D_1..D_K\} \setminus D_k\right)$
 3: if   $|D_{\text{agree}}| == |D_{\text{potential}}|$ then
 4:         $D'_k = D_{\text{agree}}$
 5: else
 6:         $D_{\text{uncertain}} = D_{\text{potential}} - D_{\text{agree}}$
 7:         $n_{\text{in}} = r(T) \cdot |D_{\text{uncertain}}|$
 8:         $D_{\text{in}} = \text{random\_sample}\left(D_{\text{uncertain}}, n_{\text{in}}\right)$
 9:         $D'_k = D_{\text{agree}} + D_{\text{in}}$
10: Return $D'_k$;
```

In this section, we conduct both synthetic and real data experiments on various datasets to validate the proposed method. The synthetic experiment follows the standard setting in literature using the benchmark data from vision recognition problems [36, 82, 89]. The real data experiment

Data	# training	# testing	# class	image size
CIFAR-10	50,000	10,000	10	32x32
CIFAR-100	50,000	10,000	100	32x32

Table 3.1: Description of datasets in synthetic experiment.



(a) symmetric 50%   (b) pairflip 45%

Figure 3.4: Noise examples for a 5-class problem.

is focused on cancer drug discovery using large-scale bioinformatics data from multiple sources [105, 124].

3.3 Experiment

3.3.1 Synthetic Experiment on Image Benchmark

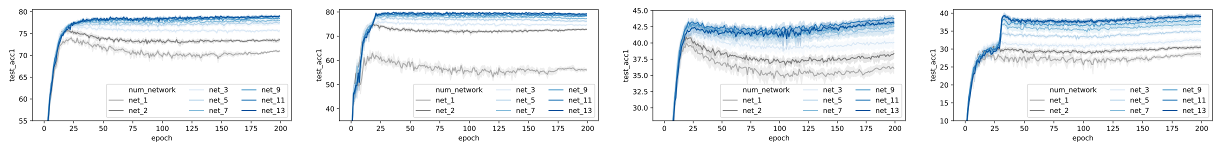
We demonstrate the effectiveness of RCL from following aspects: the benefit brought by agreement and disagreement, respectively; sensitivity analysis of vital hyper-parameters of RCL; a variant of RCL that significantly reduces computational burden and time complexity.

Datasets. We use CIFAR-10 and CIFAR-100 datasets and the description is shown in Table 3.1. The original data is clean, and we manually create corrupted labels following the strategy in [40, 117]. Two common noise scenarios are considered, symmetric flip (SYM) and pair flip (PF), as shown in Fig. 3.4. For SYM, the label of each class is uniformly random flipped to the rest classes with equal probability; for PF, the label of each class only flips to one different but similar class. The noise rate ϵ quantifies the overall proportion of labels that are flipped for each class.

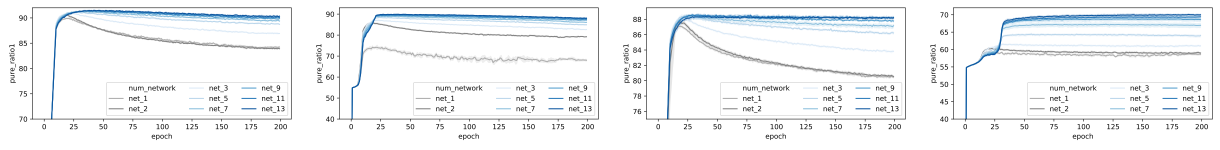
Network Architecture. We follow the same 9-layer CNN architecture as [40, 61] which is

32×32 RGB Image
3×3 conv, 128 LReLU
3×3 conv, 128 LReLU
3×3 conv, 128 LReLU
2×2 max-pool, stride 2, dropout $p = 0.25$
3×3 conv, 256 LReLU
3×3 conv, 256 LReLU
3×3 conv, 256 LReLU
2×2 max-pool, stride 2, dropout $p = 0.25$
3×3 conv, 512 LReLU
3×3 conv, 256 LReLU
3×3 conv, 128 LReLU
avg-pool
dense $128 \rightarrow 10$

Table 3.2: CNN architecture. The negative slope for each LeakyReLU is set as 0.01.



(a) CIFAR10 SYM 50% (b) CIFAR10 PF 45% (c) CIFAR100 SYM 50% (d) CIFAR100 PF 45%



(e) CIFAR10 SYM 50% (f) CIFAR10 PF 45% (g) CIFAR100 SYM 50% (h) CIFAR100 PF 45%

Figure 3.5: Episode curve (upper: test accuracy, lower: pure ratio) w.r.t. training epochs. The grey lines are SPL (net_1) and co-teaching (net_2), and blue lines the RCL. The shaded area represents the variation across 5 random seeds.

commonly used in weakly supervised learning (Table 3.2). We use Adam optimizer with a momentum of 0.9 and an initial learning rate of 0.001. The batch size is set to 128 and the maximum epoch is 200. We implemented the models using PyTorch and all the experiments are conducted on NIVIDIA GPUs. We ensure that for the same dataset and noise scenario, different comparison methods run on the same machine.

Method	Standard	SPL	De-CP	Co-T	K=3	K=5	K=7	K=9	K=11	K=13	+R	p-val	# nets
Data Noise						Test Accuracy							
CIFAR10 SYM 50%	48.52	70.92	45.53	73.45	75.72	77.44	78.01	78.47	**78.91**	78.91	**7.43**	2e-9	11
CIFAR10 PF 45%	48.65	56.08	49.24	72.77	74.59	76.28	77.28	78.25	78.70	**79.15**	**8.77**	<1e-9	13
CIFAR100 SYM 50%	21	36.21	17.51	38.14	40.05	41.83	42.43	42.96	**43.77**	43.14	**14.75**	4e-07	11
CIFAR100 PF 45%	29.57	28.63	26.17	30.44	32.51	34.90	36.86	37.85	39.02	**39.15**	**28.58**	4e-09	13
Data Noise						Pure Ratio							
CIFAR10 SYM 50%	50.31	84.22	40.48	83.95	86.96	88.82	89.47	89.86	90.11	**90.33**	**7.33**	<1e-9	11
CIFAR10 PF 45%	54.89	68.09	51.23	79.25	82.72	84.99	86.16	87.17	87.69	**88.11**	**11.18**	<1e-9	13
CIFAR100 SYM 50%	49.95	80.51	42.89	80.65	83.85	86.20	87.14	87.82	**88.20**	88.20	**9.36**	<1e-9	11
CIFAR100 PF 45%	54.84	58.66	53.42	59.03	61.07	63.94	66.97	68.65	69.36	**69.99**	**18.57**	<1e-9	13

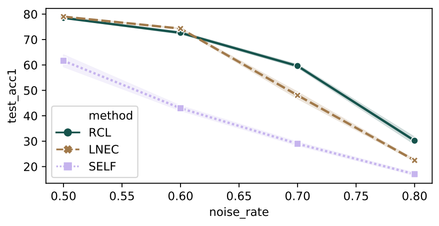
Table 3.3: Performance of non-ensemble baselines and the proposed method RCL over different number of networks (K) on fixed noise rates. Each method is repeated for 5 random seeds with average performance presented. Significance t-tests (one-side) are conducted between RCL and the best baseline. A p-value less than 0.05 is considered as significant difference.

Experimental Setup We keep the major hyper-parameter of co-teaching [40], i.e., T_{cut} shown in Alg. 5.1, and fix those that have subtle effect on results in the original paper since RCL also induces extra hyper-parameters, which are our main focus. In our experiments, we first tune $T_{\text{cut}} = \{5, 10, 15\}$ for one and two networks (i.e., SPL and Co-teaching). Then we use the best for further tuning RCL. For other hyper-parameters, we fix $\alpha = 2$ for all scenarios and use fixed β for a given noise scenario, based on the sensitivity analysis using $\beta = \{0.0, 0.1, 0.3, 0.5, 1.0, 2.0, 8.0\}$. We test over a range of number of networks $K = \{3, 5, 7, 9, 11, 13\}$, and noise rates $\epsilon = \{0.25, 0.35, 0.45\}$ for pairflip noise and $\epsilon = \{0.5, 0.6, 0.7, 0.8\}$ for symmetric noise. For all the comparison methods, we run 5 random seeds to get an average performance unless stated otherwise. And we found that the variance of RCL is relatively small across different random seeds.

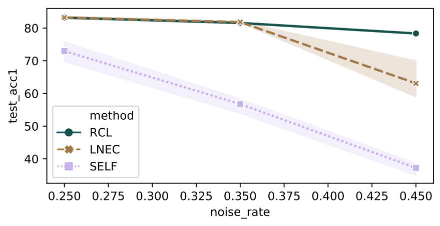
Baselines. We consider the following baselines. (1) Standard, a single network which is trained on the entire dataset. (2) SPL [50], a single network that produces curriculum based on its own knowledge. (3) Decoupling (De-CP) [68], a double-net system in which the networks only updates parameters from data whose prediction label is disagreed between two networks. (4) Co-teaching [40], a double-net system in which the two networks exchange curriculum at each iteration. (5) Ensemble consensus [63], specifically the LNEC variant, a multi-net system that explores agreement between multiple networks. (6) Self-ensemble (SELF) [77], which explores agreement between consecutive epochs within a network. Note that for the ensemble baseline SELF, the original implementation involves other hybrid components and the authors did not release the code, which makes direct comparison difficult, so we adopt the core idea of their paper which is the temporal ensemble and implement the method. Test accuracy and pure ratio serve as the evaluation metrics. The test accuracy is evaluated on *clean* test set. The pure ratio measures the average proportion of clean data that is selected by the algorithm across all mini-batches in one epoch during training. All metrics are evaluated on *one* network. For methods that contain more than one network, we only evaluate the performance of the 1st network for a fair comparison. We also find that the variation among networks are small in the end.

Benefit of agreement. The benefit of agreement comes from ensemble of multiple networks on the selection of clean samples, which can be verified by adding number of networks while fixing other components. Fig. 3.5 shows the episode curve for RCL over different number of networks for a given noise rate. We can see that the test accuracy gradually improves as the number of networks increases. For certain noisy scenarios, e.g., CIFAR100 symmetric 50%, the test accuracy reaches plateau and starts to decrease at 13 networks. For other scenarios, it continues to increase as number of network grows. Higher pure ratio generally leads to higher test accuracy. We calculate the final performance as the average of last ten epochs following [40] and then average over multiple runs. The results of RCL and baselines are shown in Table 3.3. Importantly, RCL selects significantly more clean data compared to baselines which verifies the benefit of agreement during the learning process.

Benefit of disagreement. The agreement strategy is relatively intuitive since it ensembles predictions from multiple networks after each leaner has achieved certain accuracy, however, agreement may not work when the noise rate is large since it also ensembles the error, especially during the early training stage. In such situation, disagreement plays the role that reduces the noise in gradients and helps pick out more clean samples. The disagreement takes place in terms of two levels: the first level is to exchange data in a learn-from-the-other way such that each network receives different data from the its *Peer* system, the second level is that the strength of disagreement varies along the training procedure and can be controlled by a hyper-parameter. Both levels can improve the selection of clean samples as well as the generalization performance. To verify these, we compare RCL with several ensemble methods [63, 77] which only explore agreement among multiple networks (or multiple training epochs), and all the networks use the same set of candidates to train. Fig. 3.6 shows the test accuracy of the competing methods for different noise rates on CIFAR10. Pure ratio reveals the exact same pattern and therefore is not shown in the figure. Complete results for all the datasets and noise scenarios are presented in Table 3.4. First, we find that temporal ensemble (SELF) does not work as good as network ensembles in the provided scenarios. Second, we can see that when the noise rate is small, RCL reaches similar accuracy as the state-of-art ensemble methods; when the noise rate is large, RCL yields significantly better performance compared to the

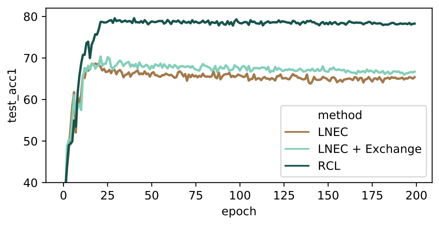


(a) CIFAR10 symmetric flip

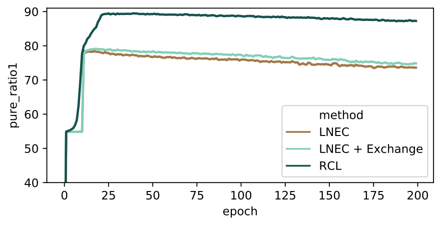


(b) CIFAR10 pair flip

Figure 3.6: Test accuracy of ensemble methods and RCL (K=9) over various noise rates on CIFAR10. The shaded area represents variation across 3 random seeds.



(a) Test accuracy



(b) Pure ratio

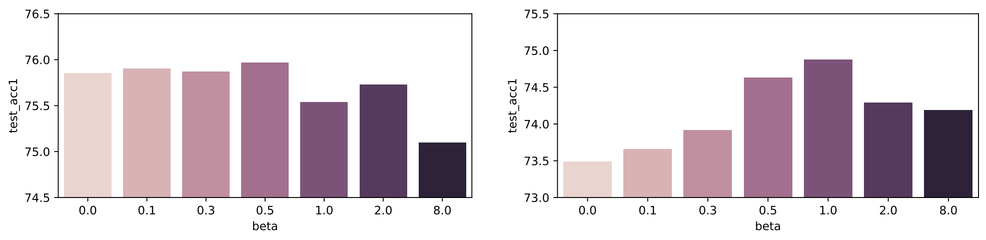
Figure 3.7: Improvement of RCL over ensemble method by each *disagreement* level on CIFAR10 pairflip 45% scenario (K=9). Average over 3 random seeds.

ensemble baselines, which demonstrates the power of introducing disagreement during the learning process. We also find that the variations of baselines are much larger compared to RCL, which indicates the robustness of the proposed method. One thing needs to point out here is that, although LNEC and RCL both use 9 networks, RCL only use the knowledge of other 8 networks, therefore it does not outperform LNEC in the easy tasks but still reaches equal goodness. Next, in order to see the improvement brought by each level of disagreement, we add the data exchange step onto LNEC baseline and compare it with pure LNEC and RCL (all use 9 networks). The result is shown in Fig. 3.7. We can see that the combination of agreement and data exchange, which makes each network receive different candidates for training, performs better than pure ensemble based on agreement (LNEC vs LNEC+Exchange). Moreover, encouraging disagreement in the early stage can further improve the selection of clean samples, leading to higher accuracy (LNEC+Exchange vs RCL).

Sensitivity analysis The hyper-parameter β controls the strength of disagreement and is important in RCL. Therefore, we would like to see (1) whether different values of this parameter will affect

Data	Noise	Test accuracy			Pure ratio		
		SELF	LNEC	CL	SELF	LNEC	CL
CF10 SYM	50%	61.65	79.00	78.51	81.38	90.07	89.85
	60%	42.97	**74.32**	72.65	64.89	**86.15**	85.14
	70%	28.99	48.05	**59.60**	46.20	61.31	**74.40**
	80%	17.04	22.45	**30.20**	29.13	31.65	**42.04**
CF10 PF	25%	72.97	83.23	83.18	88.34	93.13	92.75
	35%	56.77	81.83	81.57	75.58	91.05	90.93
	45%	37.21	63.09	**78.34**	57.61	72.22	**87.28**
CF100 SYM	50%	25.02	43.89	43.37	66.69	86.92	**87.99**
	60%	15.82	32.07	**36.12**	52.60	78.06	**82.20**
	70%	7.69	23.41	**26.71**	37.23	64.44	**72.34**
	80%	3.29	11.99	**14.79**	24.02	40.20	**49.96**
CF100 PF	25%	38.23	**52.83**	51.03	82.21	**92.04**	87.94
	35%	28.08	**46.49**	45.91	69.12	**82.28**	81.28
	45%	19.28	31.48	**38.38**	53.99	61.16	**69.17**

Table 3.4: Final performance of ensemble methods and RCL over various noise rates (K=9). Bold numbers indicates that the method is significantly better than the second best method.

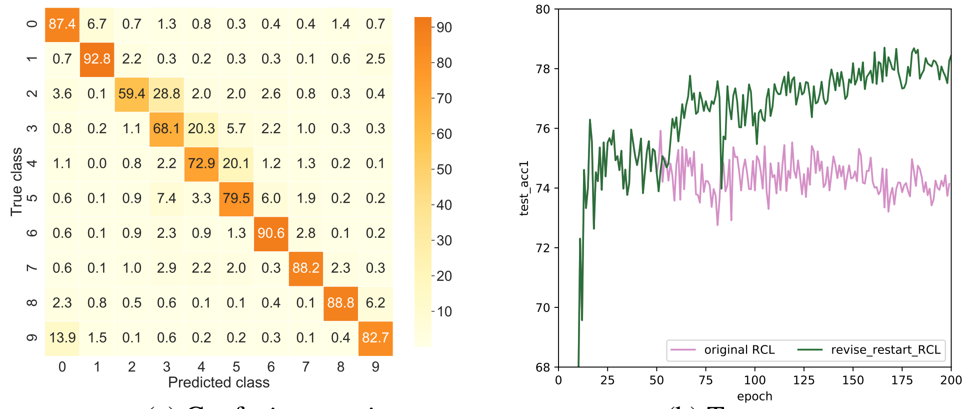


(a) symmetric 50% (b) pairflip 45%

Figure 3.8: Test accuracy of RCL over various βs, average over 3 random seeds (K=3).

the performance of RCL, and (2) how does it affect. We test over a range of different values to study its behavior on different tasks. Fig. 3.8 shows the results on CIFAR10 dataset. We can see the test accuracy shows opposite patterns on the same range of β for the two different noise scenarios. When the task is relatively easy (e.g., CIFAR10 symmetric 50%), small β yields better accuracy. If the noise rate is large, i.e., the task is more difficult, it favors relatively large β which encourages disagreement during the early stage. However, extreme large value of β is not beneficial. The difference of accuracy can reach up to 2% for different values of β while fixing all other parameters.

Reduce time complexity While being effective, RCL (as well as other methods) that involves multiple networks requires more computational power and running time compared to methods using



(a) Confusion matrix. (b) Test accuracy

Figure 3.9: Revise and restart RCL on CIFAR10 PF 45% using K=3 networks. Left: the confusion matrix after we revise the labels based on prediction. Numbers in cells denote percentages. Right: test accuracy w.r.t epochs.

one or two networks. One way to reduce time complexity without deteriorating the performance is to utilize the unselected samples during training. Therefore, we propose a revise-and-restart strategy based on the current framework. When the training reaches certain epochs (usually plateau), we first revise the labels of the unselected samples based on the current prediction of the network. We still use the first network in the system for consistency purpose, other option such as using the ensemble prediction of all networks is also applicable. Then we restart the training procedure, i.e., first introduce disagreement and then agreement. The decay factor β is reduced by half every time we restart the procedure because revising labels decreases the noise rate. Fig. 3.9 shows the corresponding result on CIFAR10 by using only 3 networks. Originally, 45% of the ground-truth labels are flipped to the adjacent class. After revising the labels of unselected samples at epoch 50, the label precision for each class is shown in Fig. 3.9a. We can see that the percentage of correct labels for each class increases significantly compared to 45%. Fig. 3.9b shows the episode curve of revise and restart compared to the original 3 networks. The test accuracies of using 3 networks are 74.1% vs 78.4% before and after. The performance of revise and restart strategy based on RCL using 3 networks reaches as high as that of using 9 networks, which is a considerable reduction on the computational burden.

In summary, we demonstrated the effectiveness of RCL on various aspects on the synthetic data.

Cell Line	VCAP	MCF7	PC3	A549	A375	HT29
Unique DP	4760	5245	5238	4195	1311	961
HQ DP	463	881	347	1244	405	127
HQ DP Test	50	100	50	100	50	30
Test size	8100	16200	8100	16200	8100	4860
Train size	763K	833K	844K	663K	206K	151K

Table 3.5: DP=Drug Profile, HQ=High Quality ($q_{dp} > 0.7$), K=1000. Small size ($\approx 10\%$) of high quality drug profiles are sampled as testing data, the rest are for training. For each cell line, a unique drug profile is associated with 162 predictable genes, therefore, sample size is # drug profile $\times$ 162.

3.3.2 Drug-induced Gene-Expression Change Prediction

Previous studies on learning with noise labels are mostly focused on vision tasks in a synthetic way. In this subsection, we apply RCL to a real-world problem in the bioinformatics domain, where the noise naturally exists in experiments due to data generation process, and demonstrates its effectiveness.

Cancer drug discovery is of high demand but also a tough problem because of low response rates and severe side effects [125]. The emerging profiling technology enables measurements of drug-induced gene-expression change (GE-change), i.e., whether the expression of a particular gene increases or decreases given a certain drug treatment, making it possible to discover new drugs and elucidate mechanism of action and toxicity of a drug candidate on the transcriptomic level. Recent years have also witnessed an increasing number of public repositories providing millions of transcriptome profiles, such as LINCS from Broad Institute [105], GEO from NCBI [10] and TCGA from NIH [123], however, large-scale profiling of drug-induced gene expression remains expensive. Therefore, there is a surging demand of computational methods for predicting drug-induced gene-expression profiles. For example, [124] proposed a deep learning framework for such tasks using drug and gene descriptors; however, due to technical and biological variations, the quality of GE-change obtained from biological experiments varies among different experiments. Due to the poor quality, half of the LINCS profiles that cost millions of dollars are discarded in regular analyses. Therefore, to make use of the full dataset, the prediction model should also take the quality of the data into consideration.

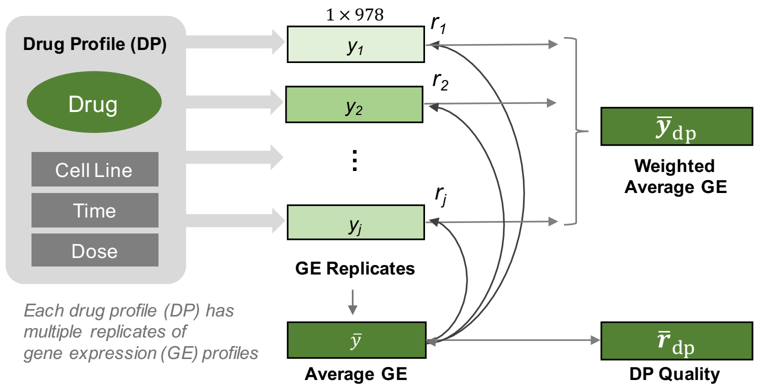


Figure 3.10: Illustration of drug profile and its quality.

Datasets. We use LINCS from Broad Institute [105] which contains 1.3 million normalized profiles and 11,000 small molecule perturbagens covering more than 70 cell lines. For each cell line, GE-change readings for 978 genes are available under different drug profiles.

Label Quality. Due to the common variation in biological experiments, in order to get reliable readings, each drug profile is repeatedly tested on 978 genes for different number of times, which results in multiple GE-change readings for the same drug profile. Some drug profiles have consistent readings while others do not, which means the label quality varies across different drug profiles. In order to conduct the experiments, we need (1) unique GE-change reading for one drug profile, (2) a rough quality measurement for generating the test set (high quality data). We do not have exact information of which drug profile is correct. In fact, no drug profile has perfectly correct or wrong GE-change readings. We can only estimate their quality through some statistical measurements. We follow [105] to estimate the quality of a drug profile and calculate the corresponding unique GE-change reading. The procedure is illustrated in Fig. 3.10. For a given drug profile (drug + cell line + time + dose), an average GE-change reading is obtained across all replicates. Then we calculate the correlation between each reading and the average reading, then take the mean, the resulted average correlation is regarded as the quality of readings for this drug profile. The final GE-change reading is obtained by weighted average of all replicates based on the correlation.

Cell	Standard		SPL		Decoupling		Co-teaching		SELF		LNEC		RCL		p-val	Best	
Line	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	K	FR
VCAP	47.64	0.462	49.36	0.473	47.70	0.473	50.26	0.482	49.23	0.474	49.12	0.472	**52.22**	**0.495**	**1e-3**	4	0.3
MCF7	51.41	0.446	55.14	0.466	54.57	0.456	56.44	0.476	56.45	0.475	54.04	0.464	**57.98**	**0.482**	**5e-4**	4	0.2
PC3	47.90	0.474	47.92	0.473	45.32	0.467	48.22	0.477	48.42	0.478	48.18	0.475	**49.04**	**0.484**	**8e-3**	4	0.1
A549	50.73	0.403	53.52	0.417	53.38	**0.422**	52.15	0.414	53.67	0.421	53.63	0.417	**54.00**	0.421	0.25	3	0.1
A375	46.42	0.396	47.14	0.395	**49.37**	**0.407**	48.87	0.399	48.43	0.402	46.95	0.393	48.99	0.395	-	3	0.4
HT29	46.39	0.441	46.51	0.444	47.18	0.449	47.86	0.456	47.12	0.453	46.66	0.452	**48.13**	**0.458**	**0.05**	3	0.3

Table 3.6: Generalization performance of comparison methods for six cell lines. K = number of networks, FR = forget rate. Each method is repeated for 5 random seeds. Significance t-tests (one-side) are conducted between RCL and the best baseline method. A p-value less than 0.05 is considered as significant difference.

Features and Label. A tuple of drug profile and gene *(drug, gene)* corresponds to a continuous reading $y_{(d,g)}$ which measures how relatively the gene expression changes compared to reference control. Therefore, we have two sets of features, drug features and gene features. In order to obtain drug features, we download SMILES (simplified molecular-input line-entry system) strings for each drug from PubChem platform[1] and use the Python package RDKit[2] to convert string representations into molecular fingerprints. The molecular fingerprint is a 1×1024 binary-coded feature where each position represent the existence or absence of a molecular substructure. For gene features, we use the Gene Ontology (GO) features preprocessed by [124] which describes the biological domain knowledge of a gene such as molecular functions and cellular components with dimension 1107. We also discretize the target $y_{(d,g)}$ into 3 classes, i.e., $y_{(d,g)} < -1.5$ (down regulate), $-1.5 \leq y_{(d,g)} \leq -1.5$ (no change), $y_{(d,g)} > 1.5$ (up regulate). The overall task is to predict the regulation effect of drug profiles on different genes.

Network Architecture. We use fully-connected layers in the shape of (2131, 128, 32, 3) neurons at each layer as the overall structure. Leaky ReLu with 0.01 slope and 0.5 drop out rate are used. We use Adam optimizer with 0.001 learning rate. The real data converges fast and we run 25 epochs and use the average over last 5 epochs as the final performance.

Experiment Set-up. We regard drug profiles that have $r_{dp} > 0.7$ as *high quality*. For each cell line, to construct training and testing dataset, we randomly sample 10% drug profiles from high quality data to serve as the testing drug profiles. The left-over mixed with remaining drug profiles are used as training drug profiles. Each drug profile is originally associated with 978 genes, however, not all genes are predictable. Therefore, we select 162 genes that are relatively predictable compared to the rest genes based on permutation tests [38]. At last, each drug profile is associated with 162 genes, and training and testing size is shown in Table 3.5. We ensure that drug profiles appear in the training set are not included in the testing set. Moreover, since a drug having significant regulation effect is rare, the data is highly imbalanced. Therefore, we down-sample the dominant class during training while keep test set unchanged. We also add macro-f1 score as an evaluation

[1] https://pubchem.ncbi.nlm.nih.gov/pc_fetch/pc_fetch.cgi
[2] https://www.rdkit.org/

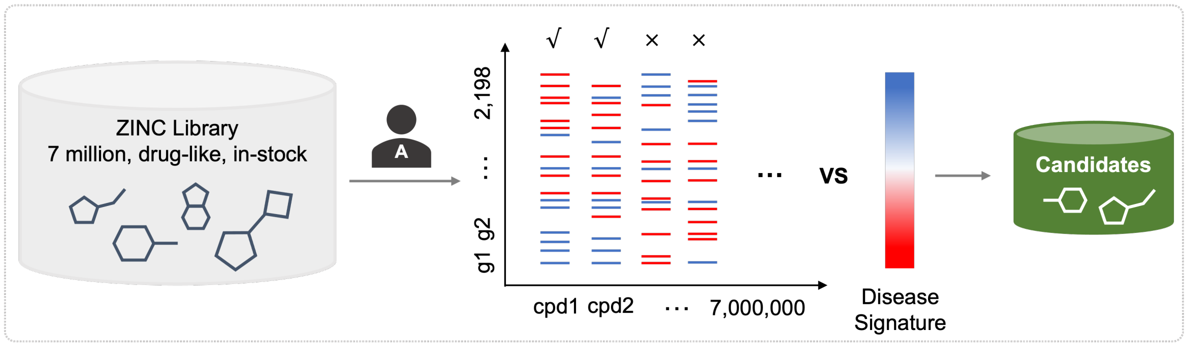


Figure 3.11: Virtual drug screening use RCL and RGES pipeline.

metric in addition to accuracy. In real experiments, we do not know the noise rate ϵ, therefore, it becomes a hyper-parameter to tune. In order to ensure that all methods eventually include the same number of data points, we first search across all possible noise (forget) rate $\{0.1, 0.2, 0.3, .., 0.9\}$ for SPL (one network) and pick the best and used for other methods. We tune $K = \{3, 4\}$ for all methods involving multiple networks and $\alpha = \{2, 3, 4\}$ and $\beta = \{2, 3, 4\}$ for RCL. Each method repeatedly run on 5 random seeds and the average is taken as the final performance.

Results. Table 3.6 shows the final performance on real data for different methods. We see that for most cell lines, RCL achieves the best accuracy compared to competing baselines. The results further confirm the effectiveness of proposed method in real-world settings, which largely broaden the potential application of RCL beyond the computer vision domain. An real drug discovery scenario is to use the RCL predictor to screen millions of compounds in the existing large library and rank compounds based on Reverse Gene Expression Score (RGES) proposed in [20], for a certain disease as illustrated in Figure 3.11.

CHAPTER 4

CONTRASTIVE LEARNING ON MOLECULAR GRAPHS WITH MULTI-LEVEL DOMAIN KNOWLEDGE

4.1 Problem Definition

A (molecular) graph can be represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, .., v_{|V|}\}$ and $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ denotes node and edge set respectively. Let $\mathbf{X} \in \mathbb{R}^{|V| \times d_1}$ be the feature matrix for all nodes in a graph, $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$ the adjacency matrix and $\mathbf{E} \in \mathbb{R}^{|\mathcal{E}| \times d_2}$ the edge features, our goal is to learn a graph encoder $\mathbf{h} = f(\mathbf{X}, \mathbf{A}, \mathbf{E}) \in \mathbb{R}^{d'}$ which maps an input graph to a vector representation without the presence of any labels. The learned encoder and representations can be used for downstream tasks directly or via finetune.

4.2 Contrastive Learning Framework

In a conventional contrastive learning framework (Fig. 5.1 left), for each graph G_i, two augmentation operators t_1 and t_2 are sampled from the family of all operators $\mathcal{T}$, and applied to G_i to obtain two correlated views $G_i^1 = t_1(G_i)$ and $G_i^2 = t_2(G_i)$. We use numbers in the **superscript** to represent different **views** throughout the paper. The correlated views are fed into a graph encoder f, producing graph representations $\mathbf{h}_i^1$ and $\mathbf{h}_i^2$, which are then mapped into an embedding space by a projection head g, yielding $\mathbf{z}_i^1$ and $\mathbf{z}_i^2$. The goal is to maximize the mutual information between the two correlated views in the embedding space via Eq (4.1).

$$\mathcal{L}^{\text{local}} = \frac{1}{n} \sum\nolimits_{i=1}^{n} \mathcal{L}_i^{\text{local}}, \tag{4.1}$$

and the loss for each sample $\mathcal{L}_i^{\text{local}}$ can be written as:

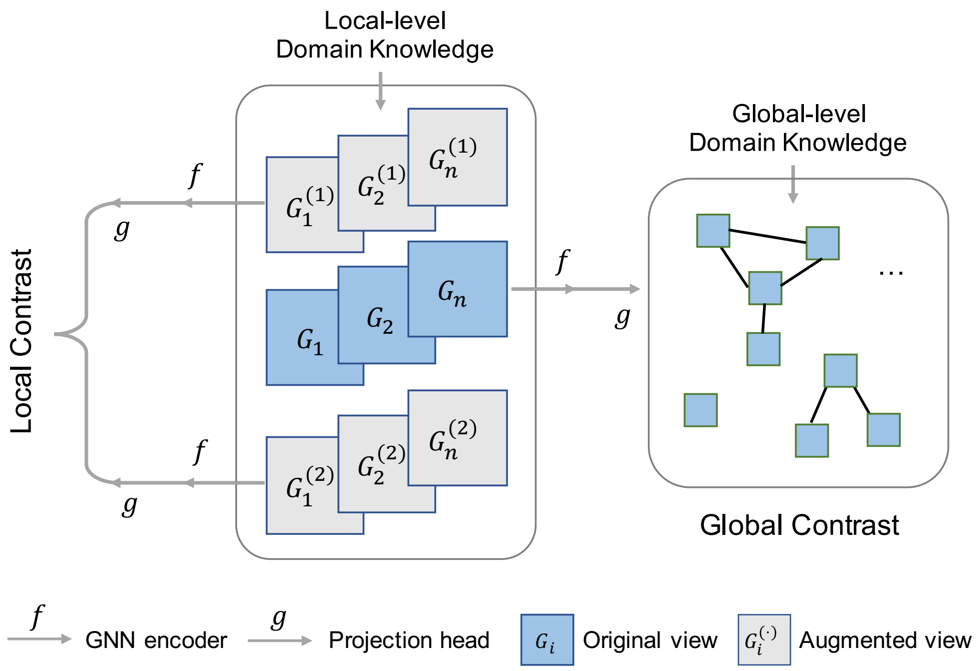


Figure 4.1: Overall framework of MoCL. First, two augmented views are generated from local-level domain knowledge. Then, together with the original view, they are fed into the GNN encoder and projection head. The local-level contrast maximizes the MI between two augmented views, while the global-level contrast maximizes the MI between two similar graphs. The similarity information is derived from global-level domain knowledge (MI: mutual information).

$$\begin{aligned}\mathcal{L}_i^{\text{local}} &= \mathcal{L}_i^1 + \mathcal{L}_i^2 \\ &= -\log \underbrace{\frac{e^{s(\mathbf{z}_i^1,\mathbf{z}_i^2)/\tau}}{\sum_{j=1,j\neq i}^{n} e^{s(\mathbf{z}_i^1,\mathbf{z}_j^2)/\tau}}}_{\text{view 1 contrast view 2}} - \log \underbrace{\frac{e^{s(\mathbf{z}_i^2,\mathbf{z}_i^1)/\tau}}{\sum_{j=1,j\neq i}^{n} e^{s(\mathbf{z}_i^2,\mathbf{z}_j^1)/\tau}}}_{\text{view 2 contrast view 1}},\end{aligned} \tag{4.2}$$

where n is the batch size, $s(\cdot,\cdot)$ is a function which measures the similarity of the two embeddings, τ is a scale parameter. The two correlated views $\mathbf{z}_i^1$ and $\mathbf{z}_i^2$ are regarded as positive pair while the rest pairs in the batch are regarded as negative pairs. The objective aims to increase the probability of occurrences of positive pairs as opposed to negative ones. Note that the negative pairs can be formed in two directions. If $\mathbf{z}_i^1$ is the anchor, all $\mathbf{z}_j^2$ in view 2 are contrasted; if $\mathbf{z}_i^2$ is the anchor, all $\mathbf{z}_j^1$ in view 1 are contrasted. Thus the loss for each sample consists of two parts as showed in Eq (4.2).

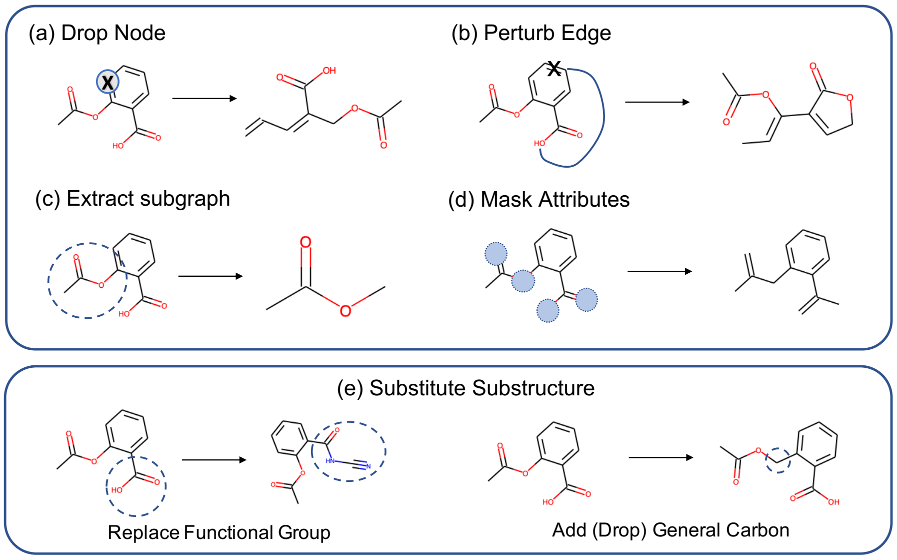


Figure 4.2: Augmentation comparison. Upper: conventional augmentations that may alter the graph semantics. Lower: proposed augmentation in which valid substructures are replaced by bioisosteres that share similar properties.

4.2.1 Local-level Domain Knowledge

Most existing approaches adopt random corruption during augmentation. For example, [141] proposed four types of augmentations for general graphs (Fig. 4.2 upper). However, such random corruption may alter the semantics of molecular graphs. For node dropping and edge perturbation, the resulting molecule is rarely biologically proper (Fig. 4.2ab). For example, dropping a carbon atom in the phenyl ring of aspirin breaks the aromatic system and results in an alkene chain (Fig. 4.2a); perturbing the connection of aspirin might introduce a five-membered lactone (Fig. 4.2b), which may drastically change the molecular properties. For subgraph extraction, the resulting structure is arbitrary and not representative for molecular functionality (Fig. 4.2c). For example, methyl acetate is a sub group of aspirin (Fig. 4.2c), but also frequently shown in many other compounds such as digitoxin and vitamin C etc. with diverse chemical structures and biological effects. Enforcing high mutual information between such augmentation pairs may produce sub-optimal representations for downstream tasks. This phenomenon has also been observed in [141] that edge perturbation deteriorates the performance of certain molecular tasks. Among the general augmentations, only attribute masking (Fig. 4.2d) does not violate the biological assumptions since it does not change the molecule, it only masks part of the atom and edge attributes.

Therefore, we aim to infuse domain knowledge to assist the augmentation process. We propose a new augmentation operator called *substructure substitution*, in which a valid substructure in a molecule is replaced by a bioisostere [71] which produces a new molecule with similar physical or chemical properties as the original one (Fig. 4.2e). We compile 218 such rules from domain resource [1]. Each rule consists of a source substructure and a target substructure represented by SMARTS string [2]. A sample rule is as follows:

```
[#6:2][#6:1](=O)[O;-,H1] >> [*:2][c:1]1nn[nH]n1
```

indicating the transition from left substructure (carboxylic acid) to the right one (nitrogen heterocycle). The substitution rules have 36 unique source substructures which can be categorized into 8 groups. We summarize the statistics of the rules in Table 4.1. Note that target substructures are all unique and different. The original 218 substitution rules mostly happen at molecular positions where heteroatoms (heavy atoms that are not C or H) and aromatic rings are presented, therefore the variation for general carbon groups is limited. Under the common assumption that changing a few general carbon atoms will not alter the molecular property too much, we add 12 additional rules to subtract and add general carbon groups from and to a molecule. Some sample rules are:

```
[*:1][CH2][CH2][*:2] >> [*:1][*:2]  (drop)
[*:1]-[*:2] >> [*:1]CC[*:2]         (add)
```

Thus, MoCL consists of 230 rules in total to generate molecule variants that share similar properties. All the rules and code are available at https://github.com/illidanlab/MoCL-DK.

Moreover, since the source substructures in the rules are very common, a molecule may contain multiple source substructures or multiple copies of the same substructure in the rule, the proposed augmentation can be applied multiple times to generate variants with much more diversity. A notable difference between proposed augmentation and general augmentation is that the proposed

[1]https://www.schrodinger.com/drug-discovery
[2]https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html

Group	# source	# target	Formula
CA	1	68	RCOO
Ester	1	7	RCOOR'
Ketone	1	15	ROR'
Phenyl	22	36	Aromatic Rings
Tbutyl	1	10	C4
dsAmide	4	18	RONR'R"
msAmide	2	32	RONR'
nsAmide	4	32	RON
Total	36	218	-

Table 4.1: Source and target statistics for substitution rules. R/R'/R" represent arbitrary carbon-containing groups.

rules are not guaranteed to be applicable to a molecule after it changes, therefore when applying proposed augmentation multiple times, we need to update the rule availability accordingly at each round. We summary the proposed augmentation procedure in Alg. 5.1.

4.2.2 Global-level Domain Knowledge

Maximizing mutual information between correlated views learns transformation-invariant representations. However, it may neglect the global semantics of the data. For example, some graphs should be closer in the embedding space since they share similar graph structures or semantics from domain knowledge. For molecular graphs, such information can be derived from multiple sources. For general graph structure, extended connectivity fingerprints (ECFPs) [91] encode the presence of substructures for molecules and are widely used to measure the structural similarity between molecular graphs. Drug-target networks [87] record the drug-protein interaction information which is one of the most informative biological activity measures. In this section, we first define graph similarity from general molecular graphs, then we propose two ways to incorporate the global semantics into our learning framework.

4.2.2.1 Similarity calculation

Given the ECFP of two molecules, $e_1, e_2 \in \{0, 1\}^m$ where m is the vector length and 1 indicates the presence of certain substructures, the similarity of e_1 and e_2 can be calculated as the Tanimoto coefficient [9]:

$$s(e_1, e_2) = \frac{N_{12}}{N_1 + N_2 - N_{12}}, \tag{4.3}$$

where N_1, N_2 denotes the number of 1s in e_1, e_2 respectively, and N_{12} denotes the number of 1s in the intersection of e_1, e_2. The resulted coefficient $s(e_1, e_2) \in [0, 1]$ and a larger value indicates higher structural similarity. Similarly, for drug-target network, $e_1, e_2 \in \{0, 1\}^m$ becomes the interaction profile of a drug to all proteins where m is the total number of proteins. The drug similarity can be calculated the same as Eq. (4.3).

Algorithm 4.1: Pseudocode of domain augmentation.

```
Input: Molecule graph $G$, repeat time $R$, rules $\mathcal{T}$
Output: Augmented graph $G'$.
1: for $r = 1$ to $R$ do
2:     while $\mathcal{T}$ do
3:         sample $t \sim \mathcal{T}$            # one augmentation rule
4:         $\{G^1, G^2, .., G^k\} = t(G)$     # all possible products
5:         random choose $G = G^i$
6:         update available $\mathcal{T}$ # rules may no longer be valid
7:         break;
8: Return $G'$;
```

4.2.2.2 Global-level Objective

We propose two strategies for using the global similarity information. One strategy is to use it as direct supervision. Given embeddings of two original graphs $\mathbf{z}_i$ and $\mathbf{z}_j$, we measure the similarity between them as $\theta(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^T \mathbf{z}_j}{\|\mathbf{z}_i\|\|\mathbf{z}_j\|}$. We optimize the similarity using least square loss as follows:

$$\mathcal{L}_i^{\text{global}} = \sum\nolimits_{j \neq i} \mathcal{L}_{ij}^{\text{global}} = \sum\nolimits_{j \neq i} \|\theta(\mathbf{z}_i, \mathbf{z}_j) - s_{i,j}\|_2^2,$$

where $s_{i,j}$ is the similarity from Eq. (4.3).

The second strategy is to utilize a contrastive objective in which similar graph pairs have higher mutual information as compared to the background. The objective is written as:

$$\mathcal{L}_i^{\text{global}} = -\log \frac{\sum_{j=1, j \in \mathcal{N}_i}^{n} e^{s(\mathbf{z}_i, \mathbf{z}_j)/\tau}}{\sum_{j=1, j \notin \mathcal{N}_i}^{n} e^{s(\mathbf{z}_i, \mathbf{z}_j)/\tau}},$$

where $\mathcal{N}_i$ refers the neighbors of graph i. The neighbors can be derived from global similarity by setting a threshold or a neighborhood size. The global loss for all graphs thus becomes:

$$\mathcal{L}^{\text{global}} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i^{\text{global}}. \tag{4.4}$$

Finally, the full objective of the proposed MoCL can be written as:

$$\mathcal{L} = \mathcal{L}^{\text{local}} + \lambda \mathcal{L}^{\text{global}}, \tag{4.5}$$

where λ is a tuning parameter that controls the emphasis between local loss and global loss. We summarize the pseudo code of the entire framework in Alg. 4.2.

Algorithm 4.2: Pseudocode of proposed framework.

Input: Molecule graphs G, rules $\mathcal{T}$, hyper parameter λ, number of epochs M.
Output: Graph encoder f.

```
1: for $m = 1$ to $M$ do
2:     for $iter = 1$ to max_iter do
3:         $G^1 = \text{Alg.1}(G, \mathcal{T}), G^2 = \text{Alg.1}(G, \mathcal{T})$
4:         $\mathbf{h}^1 = f(G^1), \mathbf{h}^2 = f(G^2), \mathbf{h} = f(G)$
5:         $\mathbf{z}^1 = g(\mathbf{h}^1), \mathbf{z}^2 = g(\mathbf{h}^2), \mathbf{z} = g(\mathbf{h})$
6:         Calculate local loss by Eq. (4.1)
7:         Calculate global loss by Eq. (4.4)
8:         Optimize $f$ and $g$ using Eq. (4.5)
9: Return $f$;
```

4.2.3 Connection to Metric Learning

It has been well studied that optimizing objective Eq. (4.1) is equivalent to maximizing a lower bound of the mutual information between the correlated views, also a lower bound of the mutual information between input and the hidden representations [80, 23]. Formally, denote $\mathbf{Z}^1$ and $\mathbf{Z}^2$

as the random variables for the embeddings of augmentations, $\mathbf{X}$ the variable for original input features:

$$\mathcal{L}^{\text{local}} \leq I(\mathbf{Z}^1; \mathbf{Z}^2) \leq I(\mathbf{X}; \mathbf{Z}^1, \mathbf{Z}^2).$$

Beyond mutual information maximization, in this section, we provide additional justification for the proposed method from the perspective of metric learning, which unifies the local and global objectives. We show the following important result:

Lemma 2 *Assume the projection head g is an identity mapping, i.e., $\mathbf{z} = g(\mathbf{h}) = \mathbf{h}$, and the similarity function $s(\cdot,\cdot)$ is inner product, i.e., $s(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T \mathbf{z}_j$. Consider 1-nearest neighbor of each graph in the batch for global structure information, and $\lambda = 1$, the objective $\mathcal{L}_i$ is equivalent to the following:*

$$\mathcal{L}_i \propto \sum_{j \neq i} \underbrace{\|\mathbf{z}_i^1 - \mathbf{z}_i^2\|^2 - \|\mathbf{z}_i^1 - \mathbf{z}_j^2\|^2}_{\textit{local contrast view 1}} + \underbrace{\|\mathbf{z}_i^2 - \mathbf{z}_i^1\|^2 - \|\mathbf{z}_i^2 - \mathbf{z}_j^1\|^2}_{\textit{local contrast view 2}}$$
$$+ \sum_{j \neq k, k \in \mathcal{N}_i} \underbrace{\|\mathbf{z}_i - \mathbf{z}_k\|^2 - \|\mathbf{z}_i - \mathbf{z}_j\|^2}_{\textit{global contrast}} + Const.$$

Proof:

$$\begin{aligned}
\mathcal{L}_i &= \log \frac{\sum_{j\neq i}^n e^{s(\mathbf{z}_i^1, \mathbf{z}_j^2)}}{e^{s(\mathbf{z}_i^1, \mathbf{z}_i^2)/\tau}} + \log \frac{\sum_{j\neq i}^n e^{s(\mathbf{z}_i^2, \mathbf{z}_j^1)/\tau}}{e^{s(\mathbf{z}_i^2, \mathbf{z}_i^1)/\tau}} + \log \frac{\sum_{j\neq k, k\in\mathcal{N}_i}^n e^{s(\mathbf{z}_i, \mathbf{z}_j)/\tau}}{e^{s(\mathbf{z}_i, \mathbf{z}_k)}} \\
&= \log \sum_{j\neq i}^n e^{s(\mathbf{z}_i^1, \mathbf{z}_j^2)/\tau - s(\mathbf{z}_i^1, \mathbf{z}_i^2)/\tau} + \log \sum_{j\neq i}^n e^{s(\mathbf{z}_i^2, \mathbf{z}_j^1)/\tau - s(\mathbf{z}_i^2, \mathbf{z}_i^1)/\tau} + \log \sum_{j\neq k, k\in\mathcal{N}_i}^n e^{s(\mathbf{z}_i, \mathbf{z}_j)/\tau - s(\mathbf{z}_i, \mathbf{z}_k)/\tau}
\end{aligned}$$

By applying first-order Taylor expansion we have:

$$
\begin{aligned}
\mathcal{L}_i &\approx \sum_{j\neq i}^{n} e^{s(\mathbf{z}_i^1,\mathbf{z}_j^2)/\tau - s(\mathbf{z}_i^1,\mathbf{z}_i^2)/\tau} + \sum_{j\neq i}^{n} e^{s(\mathbf{z}_i^2,\mathbf{z}_j^1)/\tau - s(\mathbf{z}_i^2,\mathbf{z}_i^1)/\tau} + \sum_{j\neq k, k\in\mathcal{N}_i}^{n} e^{s(\mathbf{z}_i,\mathbf{z}_j)/\tau - s(\mathbf{z}_i,\mathbf{z}_k)/\tau} - 3 \\
&\approx \frac{1}{\tau}\Big[\sum_{j\neq i}^{n} s(\mathbf{z}_i^1,\mathbf{z}_j^2) - s(\mathbf{z}_i^1,\mathbf{z}_i^2) + \sum_{j\neq i}^{n} s(\mathbf{z}_i^2,\mathbf{z}_j^1) - s(\mathbf{z}_i^2,\mathbf{z}_i^1) + \sum_{j\neq k, k\in\mathcal{N}_i}^{n} s(\mathbf{z}_i,\mathbf{z}_j) - s(\mathbf{z}_i,\mathbf{z}_k)\Big] - 3 \\
&= \frac{1}{\tau}\Big[\sum_{j\neq i}^{n} \mathbf{z}_i^{1T}\mathbf{z}_j^2 - \mathbf{z}_i^{1T}\mathbf{z}_i^2 + \sum_{j\neq i}^{n} \mathbf{z}_i^{2T}\mathbf{z}_j^1 - \mathbf{z}_i^{2T}\mathbf{z}_i^1 + \sum_{j\neq k, k\in\mathcal{N}_i}^{n} \mathbf{z}_i^T\mathbf{z}_j - \mathbf{z}_i^T\mathbf{z}_k\Big] - 3 \\
&= \frac{1}{2\tau}\Big[\sum_{j\neq i}^{n} \|\mathbf{z}_i^1 - \mathbf{z}_i^2\|^2 - \|\mathbf{z}_i^1 - \mathbf{z}_j^2\|^2 + \|\mathbf{z}_i^2 - \mathbf{z}_i^1\|^2 - \|\mathbf{z}_i^2 - \mathbf{z}_j^1\|^2 \\
&+ \sum_{j\neq k, k\in\mathcal{N}_i}^{n} \|\mathbf{z}_i - \mathbf{z}_k\|^2 - \|\mathbf{z}_i - \mathbf{z}_j\|^2\Big] - 3 \\
&\propto \sum_{j\neq i} \|\mathbf{z}_i^1 - \mathbf{z}_i^2\|^2 - \|\mathbf{z}_i^1 - \mathbf{z}_j^2\|^2 + \|\mathbf{z}_i^2 - \mathbf{z}_i^1\|^2 - \|\mathbf{z}_i^2 - \mathbf{z}_j^1\|^2 \\
&+ \sum_{j\neq k, k\in\mathcal{N}_i} \|\mathbf{z}_i - \mathbf{z}_k\|^2 - \|\mathbf{z}_i - \mathbf{z}_j\|^2 - 6\tau
\end{aligned}
$$

The lemma above connects the objective design to the metric learning. The equation consists of three triplet losses [19] which corresponds to the two local losses and the global loss respectively. As such, the MoCL objective aims to pull close the positive pairs while pushing away the negative pairs from both local and global perspective. Detailed proofs can be found in Appendix.

4.3 Experiment

In this section, we conduct extensive experiments to demonstrate the proposed method by answering the following questions:

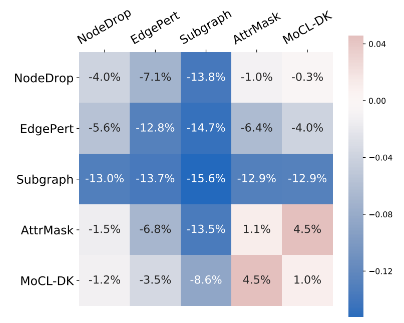
Q1. Does local-level domain knowledge (MoCL-DK) learns better representations than general augmentations? How does combination of different augmentations behave?

Q2. Does global-level domain knowledge (MoCL-DK-G) further improves the learned representations? Do the two proposed global losses perform the same?

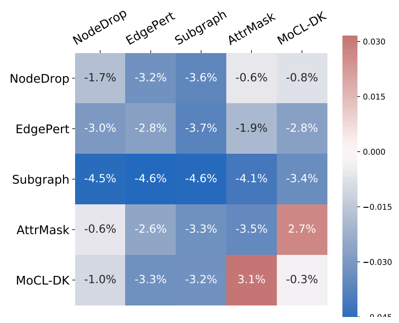
Q3. How does the hyper-parameters (λ, neighbor size) involved in MoCL affect the model performance?

Dataset	# Tasks	Size	Avg. Node	Avg. Degree
bace	1	1513	34.1	36.9
bbbp	1	2050	23.9	25.8
clintox	2	1483	26.1	27.8
mutag	1	188	17.8	19.6
sider	27	1427	33.6	35.4
tox21	12	7831	18.6	19.3
toxcast	617	8597	18.7	19.2

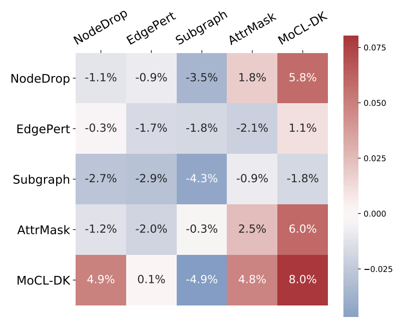
Table 4.2: Basic statistics for all datasets



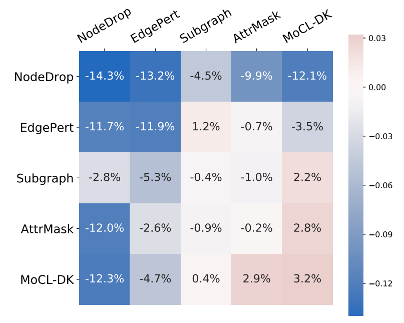
(a) bace



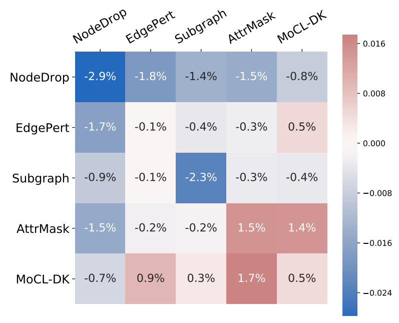
(b) bbbp



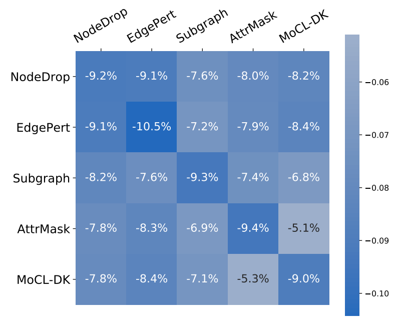
(c) clintox



(d) mutag



(e) sider



(f) tox21

Figure 4.3: Augmentation combination under linear evaluation protocol. Each cell represents the performance difference between a vanilla GNN trained from scratch (upper-bound) and learned representations (fixed) from a pretrained model plus a linear classifier under a given augmentation combination. Each number is averaged from 5 runs. Blue represents negative value and red positive. Higher value is better. MoCL-DK is the proposed augmentation with local-level domain knowledge.

4.3.1 Evaluation Protocols

The evaluation process follows two steps, we first pretrain a model based on any comparison method, and then evaluate the learned model on downstream tasks. We adopt two evaluation protocols:

- Linear protocol: fix the representation from pretrained model and finetune a linear classifier on top of it.
- Semi-supervised protocol: sample a small set of labels of the downstream task and use the weights of learned graph encoder as initialization meanwhile finetune all the layers.

which are most commonly used in literature [109, 43, 134, 141].

4.3.2 Experimental Setup

Datasets and Features. We use 7 benchmark molecular datasets in the literature [43, 134, 109] to perform the experiments, which covers a wide range of molecular tasks such as binding affinity, response to bioassays, toxicity and adverse reactions:

- bace [106]: a dataset containing the binding results between molecules and human proteins .
- bbbp [69]: a dataset measuring the blood-brain barrier penetration property of molecules.
- mutag [93]: a dataset recording the mutagenic effect of a molecule on a specific gram negative bacterium.
- clintox & tox21 & toxcast [33, 79, 90]: datasets that contains the molecule toxicity from FDA clinical trials (clintox) and in vitro high-throughput screening (tox21 and toxcast).
- sider [58]: a dataset containing the adverse drug reactions (ADR) of FDA approved drugs.

The basic statistics of the datasets (size, tasks, molecule statistics) are summarized in Table 4.2. In this paper, we mainly focus on classification tasks as prior works [43, 134, 109], therefore we use AUC [113] as the major evaluation metric.

For molecular graphs, we use both atom features and bond features as inputs. We use i) atomic number and ii) chirality tag as features for atoms and i) bond type and ii) bond directions as features for chemical bonds [43].

Model Architectures. We use GIN [129] as our graph encoder f which has been shown to be the most expressive graph neural network layer in prior works [43]. It also allows us to incorporate edge features of molecules into the learning process. The update rule for each GIN layer can be written as:

$$\mathbf{x}_i^{l+1} = \mathrm{MLP}_\theta\left(\mathbf{x}_i^l + \sum\nolimits_{j \in \mathcal{N}_i} \mathrm{ReLU}\left(\mathbf{x}_j^l + \mathbf{e}_{j,i}\right)\right),$$

where $\mathbf{x}_i^l$ is the node representation at l-th layer, $\mathcal{N}_i$ denotes the neighbor nodes of i-th node and $\mathbf{e}_{j,i}$ represents the edge feature between node i and j. MLP_θ is a two-layer perceptron parameterized by θ. Note that MLP here is for a single GIN layer in order to make the GIN layer the most expressive. After obtaining the node representations for all atoms in a molecule, we average them to get the graph representation $\mathbf{h}$.

We use another two-layer perceptron for the projection head g in our framework following literature [21, 134]. It has been shown that a projection head with nonlinear transformation is necessary for a better representation of the layer before it due to information loss in the contrastive learning loss [21]. After adding a projection head, the representations at previous layer, ie., $\mathbf{h}$, can benefit more for downstream tasks. We use cosine similarity for the critic function $s(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^T \mathbf{z}_j / \|\mathbf{z}_i\| \|\mathbf{z}_j\|$ [134].

Baselines For both linear and semi-supervised evaluation protocols, we adopt the following three types of baselines for comparison:

- Vanilla GNN (Scratch): train a standard nonlinear GNN model on labeled data of the downstream task.

- General GNN self-supervised learning or pretraining baselines: i) InfoGraph [109], which maximizes the mutual information between nodes and graph; ii) Edge Pred & Context Pred [43]: which uses the node embeddings to predict graph edge and neighbor context in order to learn meaningful node representations; iii) Masking [43]: which masks the atom attributes and tries to predict them.

- Graph contrastive learning baselines: we adopt the four types of general augmentations for graph in [134]: i) node dropping; ii) edge perturbation; iii) subgraph extraction; iv) attribute masking for comparison.

Protocol	Linear Protocol							Semi-supervised Protocol						
Method | Dataset	bace	bbbp	clintox	mutag	sider	tox21	toxcast	bace	bbbp	clintox	mutag	sider	tox21	toxcast
scratch	0.785	0.861	0.647	0.918	0.606	0.820	0.710	0.525	0.695	0.494	0.803	0.552	0.670	0.530
graphInfomax	0.594	0.611	0.458	0.771	0.502	0.615	0.562	0.614	0.735	0.487	0.887	0.523	0.589	0.535
contextpred	0.522	0.724	0.506	0.819	0.498	0.554	0.542	0.566	0.731	0.502	0.846	0.525	0.659	0.514
edgepred	0.662	0.592	0.504	0.622	0.502	0.500	0.501	0.604	0.694	0.486	0.915	0.545	0.615	0.529
masking	0.678	0.764	0.581	0.826	0.566	0.722	0.617	0.621	**0.776**	0.585	0.879	**0.551**	0.640	0.538
drop_node	0.746	0.843	0.635	0.775	0.577	0.728	0.633	0.603	0.767	0.492	0.836	0.542	0.656	0.525
perturb_edge	0.657	0.833	0.630	0.799	0.605	0.715	0.619	0.527	0.748	0.516	**0.938**	0.547	0.629	0.516
subgraph	0.629	0.815	0.603	0.914	0.583	0.727	0.625	0.565	0.769	0.539	0.918	0.548	0.656	0.514
mask_attributes	0.796	0.826	0.671	0.916	0.621	0.726	0.623	0.622	0.710	0.478	0.897	0.549	**0.666**	0.543
MoCL-DK	0.801	0.870	**0.727**	**0.950**	0.615	0.740	0.636	**0.650**	0.765	**0.588**	0.903	0.546	0.645	**0.539**
MoCL+AttrMask	**0.831**	**0.892**	0.695	0.947	**0.623**	**0.768**	**0.653**	0.630	0.748	0.549	0.909	0.536	0.661	0.536
MoCL-DK-G(LS)	0.831	0.892	0.724	0.958	0.623	**0.777***	**0.659***	0.662	0.766	0.623	0.907	0.558	0.666	**0.547***
MoCL-DK-G(CL)	**0.845***	0.905	**0.750***	**0.969***	**0.628***	0.768	0.653	**0.706***	**0.809***	**0.623***	0.916	0.565	0.686	0.546
MoCL+AttrMask-G(CL)	0.833	**0.911***	0.747	0.962	0.625	0.774	0.654	0.695	0.806	0.618	0.913	**0.567***	**0.687***	0.544

Table 4.3: Averaged test AUC of comparison methods under linear and semi-supervised protocol (5 runs). **Bold** number denotes the best performance for local-level (augmentation) comparison. **Bold*** number denotes the best performance after incorporating global similarity information (MoCL-G). LS and CL represents least-square and contrastive global loss, respectively.

Implementation Details. We use 3 layers of GIN for all methods since 3-hops neighborhood covers most aromatic rings and is usually sufficient for molecular structure learning [91]. The dimensions for GIN layer and embedding layer are 512 and 128 respectively. We use Adam as optimizer with initial learning rate of 0.001 for all methods. We use dropout ratio 0.5 for GIN layers and default settings for baselines. The batch size is 32 across all scenarios. For pretraining models, the running epoch is fixed to 100. For downstream tasks, we use early stop via validation set. We implement all models using Pytorch [81] and run them on Tesla K80 GPUs.

The variation of results for a dataset comes from two sources, the pretrained model and the downstream task. By comparing them, we find the variation of pretrained model (by applying different seeds) is much smaller than the variation of downstream task (by different training-testing splits). Therefore, for each dataset, we use its molecular graphs to pretrain a model (1 seed) and then apply it to downstream task on the same dataset using different splits (5 seeds). We do not evaluate transfer learning setting in this paper where a pretrained model is applied to another dataset. During downstream task, we split the dataset into training (0.8), validation (0.1) and testing (0.1) set, we use validation set for early stop and evaluate the AUC on testing set. For semi-supervised protocol where only a small fraction of labels is used to train, since the data sizes are different, the ratio is picked from {0.01, 0.05, 0.5} such that around 100 molecules being selected for each dataset. For local-level domain knowledge, we use augmentation ratio 0.2 for general augmentations as prior work [134] and different augmentation times $\{1, 2, 3, 5\}$ for the proposed method. For example, MoCL-DK3 denotes applying domain augmentation 3 times. For global-level domain knowledge part, we try $\lambda = \{0.5, 1.0, 5.0, 10.0\}$ and 4 different nearest neighbor sizes for each dataset based on its size. We use ECFP with dimension 1024 to calculate the global similarity.

Table 4.4 shows the detailed parameter settings for all datasets. Semi-ratio depends on the data size such that around 100 molecule labels are sampled from each dataset. The neighbor size also depends on the data size such that the number of clusters is between 5 and 30 for all datasets. The parameter λ which controls the weight between local and global loss, and augmentation time for MoCL-DK are all set to the same set of values for all datasets.

Dataset	Size	Semi-ratio	Neigbor Size	λ	DK
bace	1513	0.05	{50, 100, 150, 300}	{0.5, 1, 5, 10}	{1,2,3,5}
bbbp	2050	0.05	{50, 100, 150, 300}	{0.5, 1, 5, 10}	{1,2,3,5}
clintox	1483	0.05	{50, 100, 150, 300}	{0.5, 1, 5, 10}	{1,2,3,5}
mutag	188	0.5	{10, 20, 30, 40}	{0.5, 1, 5, 10}	{1,2,3,5}
sider	1427	0.05	{50, 100, 150, 300}	{0.5, 1, 5, 10}	{1,2,3,5}
tox21	7831	0.01	(600, 800, 1000}	{0.5, 1, 5, 10}	{1,2,3,5}
toxcast	8597	0.01	{600, 800, 1000}	{0.5, 1, 5, 10}	{1,2,3,5}

Table 4.4: Detailed experimental settings for each dataset.

Unlike prior work [134] in which only node, node features and connectivity information are used as input, our GNN incorporates edge features, therefore, the implementation of general augmentation is slightly different from [134]. We list the operations for both node (features) and edge (features) in Table 4.5.

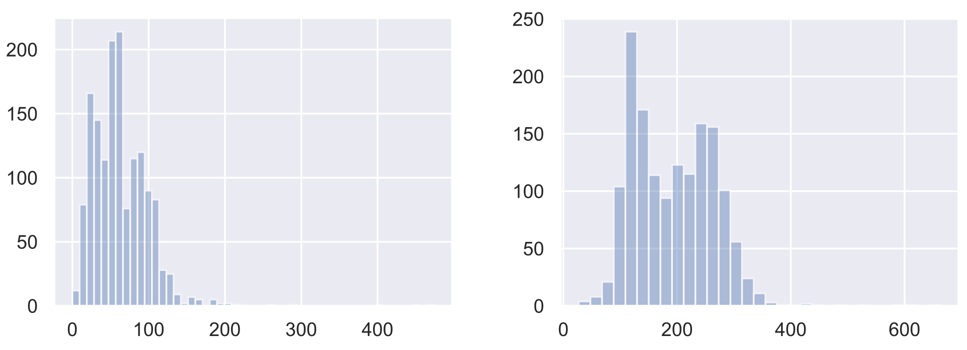
Augmentation	Node	Node features	Edge	Edge features
Drop Node	removed	removed	removed	removed
Perturb Edge	-	-	permuted	permuted
Subgraph	subsample	subsample	keep	keep
Mask Attributes	mask	mask	mask	mask

Table 4.5: Implementation details for general augmentation. Edge refers all edges that reach out from the corresponding node. - denotes no change.

4.3.3 Local-level domain knowledge (Q1)

We first examine whether the proposed augmentation helps learn a better representation. Figure 4.4 shows the distribution of number of augmentations that can be generated by applying MoCL-DK1 (left: from rules of substituting functional groups; right: from rules of adding/dropping general carbons). Other datasets reveal the same pattern therefore we do not include them due to space limit. We see that MoCL-DK1 can generate considerable number of augmentations for the molecules. If we apply MoCL-DK multiple times (MoCL-DK3, MoCL-DK5), the number of possible products can further increase drastically.

Since the contrastive framework involves two correlated views, different augmentation schemes can be applied to each view. Figure 4.3 shows the results of different augmentation combinations

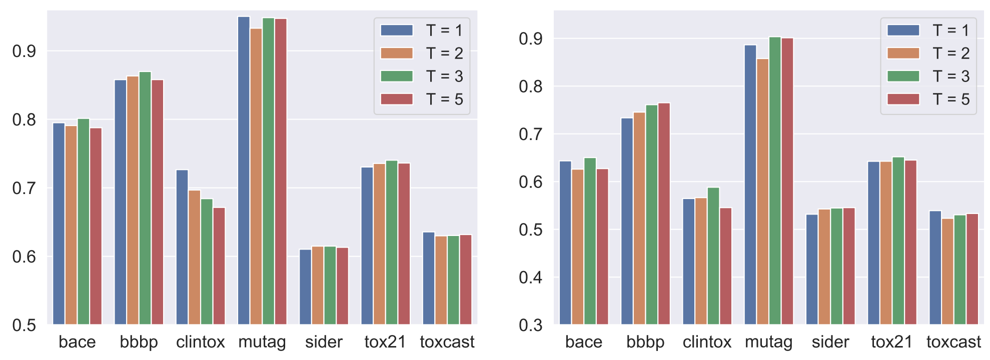


(a) Function group rule (b) General carbon rule

Figure 4.4: Distribution of augmentations that can be generated by proposed augmentation rules (dataset: bace).

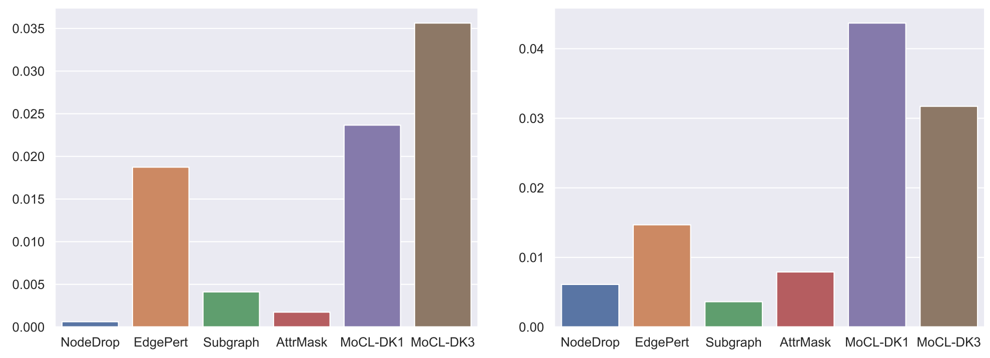
under linear protocol for all datasets (the results of toxcast is similar as tox21 therefore we remove it due to space limit). MoCL-DK represent applying domain augmentation by only once. We can see that i) the representations from MoCL-DK (diagonals) plus a linear classifier yield prediction accuracies which are on-par with a deep learning model train from scratch (bace, bbbp, sider), or even better than it (clintox, mutag). ii) the proposed augmentation MoCL-DK combined with other augmentations almost always produce better results compared to other combinations (rows and columns that contain MoCL-DK is usually higher). iii) Attribute masking and MoCL-DK are generally effective across all scenarios, combining them often yields even better performance. This verifies our previous assumption that MoCL-DK and attribute masking does not violate the biological assumption and thus works better than other augmentation. Moreover, harder contrast, e.g., combination of different augmentation schemes benefits more as compared to one augmentation schemes (MoCL-DK + AttrMask often produce the best results). This phenomenon is reasonable and also observed in prior works [134].

For semi-supervised protocol, the results are weaker, we did not include the augmentation combination figure due to space limit. But the complete results for all comparison methods for both linear and semi-supervised protocol can be found in Table 4.3. The fourth block of methods in Table 4.3 represents results for proposed augmentation. Table 4.3 also presents global results which we will mention in the next subsection.



(a) Linear protocol (b) Semi-supervised protocol

Figure 4.5: Average test AUC of MoCL-Local across different augmentation strengths (repeat times) for all datasets.



(a) Linear protocol (b) Semi-supervised protocol

Figure 4.6: Average test AUC gain from global domain knowledge for different augmentations across all datasets.

The proposed augmentation MoCL-DK can be applied multiple times to generate more complicated views. We tried over a range of different augmentation strengths and report the corresponding results for all datasets in Figure 4.5. We can see that for most datasets, as we apply more times the proposed augmentation, the performance first increases and then decreases. MoCL-DK3 usually achieves better results than others. For certain datasets (clintox, toxcast) the trend is not very clear between the two evaluation protocols.

4.3.4 Global-level domain knowledge (Q2)

We next study the role of global-level domain knowledge by examining the following sub-questions:

- Does global similarity helps general (baseline) augmentations? Does it helps the proposed augmentation? Are the effectiveness the same?
- How does different global losses behave, i.e., direct supervision as least square loss v.s. contrastive loss, across all datasets, which one is better?

Figure 4.6 shows the performance gain by incorporating global similarity information for general (baseline) augmentations and the proposed augmentation. Each bar represents the median gain across all 7 datasets for a particular augmentation scheme. we can see that global information generally improves all augmentation schemes (the bars are positive). Interestingly, the gain for proposed domain augmentation (MoCL-DK1 and MoCL-DK3) are much higher as compared to other augmentations schemes. Note that we used the same set of global-level hyper-parameters for all augmentations for fair comparison. Table 4.6 shows the performance for different global losses under both evaluation protocol. We can see that contrastive loss (CL) for the global similarity achieves better results than directly using it as supervision by least-square loss (LS).

We summarize the complete results for all comparison method in Table 4.3. We can see that i) contrastive learning works generally better than traditional graph pretraining methods, especially in linear protocol; ii) The proposed augmentation outperforms general augmentations. By combining MoCL augmentation and attribute masking, the results are even better for some datasets; iii) The global similarity information further improves the learned representations. Moreover, without combining with attribute masking, MoCL augmentation only already achieves the best performance under most scenarios after adding global information. The learned representations plus a linear classifier can achieve higher accuracy than a well-trained deep learning model. In summary, the proposed method is demonstrated to be effective for various molecular tasks.

4.3.5 Sensitivity Analysis (Q3)

Finally we check the sensitivity of global-level hyper-parameters, ie., the neighbor size and λ that controls the weight between local and global loss. Figure 4.7 shows the performance surface under

Protocol	Linear		Semi-supervised	
Dataset	LS	CL	LS	CL
bace	0.831	**0.845**	0.662	**0.701**
bbbp	0.891	**0.903**	0.766	**0.809**
clintox	0.724	**0.750**	0.608	**0.619**
mutag	0.954	**0.963**	0.895	**0.907**
clintox	0.623	**0.628**	0.551	**0.563**
tox21	0.774	0.768	0.655	**0.686**
toxcast	0.659	0.653	0.547	0.546

Table 4.6: Comparison between different global losses under MoCL-DK1 augmentation. LS: directly using global similarity and optimize by least-square loss; CL: contrastive loss using nearest neighbor derived from global similarity.

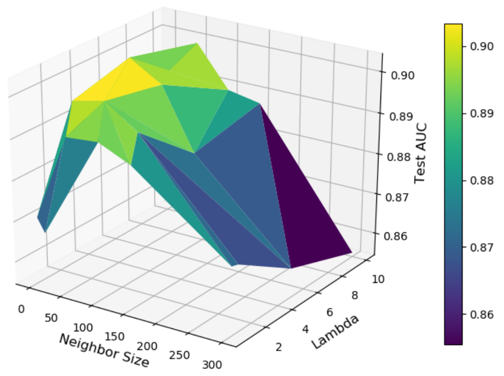


Figure 4.7: Average test AUC of different neighbor size and λ for MoCL-DK1-G under linear protocol (dataset: bbbp).

different hyper-parameter combination of the proposed method for bbbp dataset. We can see that a relatively smaller neighbor size (not too small) and larger weights (not too large) for the global loss leads to a best result. Other datasets also show the similar pattern.

4.3.6 Discussion

We provide additional observations and discussion in this subsection. First, we observe that representations which perform well under linear evaluation do not guarantee to be better in the semi-supervised setting. Since we finetune all the layers in semi-supervised learning, an overly delicate representation as initialization may not produce the best results in a fully nonlinear setting. Second, the effectiveness of contrastive learning also depends on the property of the dataset as well as the nature of the task. For example, single property prediction (mutag, bbbp) benefits more from pretraining as compared to toxicity prediction (tox21, toxcast) since it not only depends on the compound structure, but also the cellular environment. Therefore, incorporating drug-target network information may be more helpful to these datasets, which is our future direction.

CHAPTER 5

SEARCH-BASED MULTI-OBJECTIVE MOLECULAR GENERATION AND PROPERTY OPTIMIZATION

The proposed framework MolSearch as shown in Figure 5.1 consists of two search stages: a HIT-MCTS stage and a LEAD-MCTS stage. HIT-MCTS aims to modify molecules for better biological properties while LEAD-MCTS stage seeks molecules with better non-biological properties. Each stage utilizes a multi-objective Monte Carlo search tree to search for desired molecules.

5.1 Problem Definition

Molecule modification can be mathematically formulated as a Markov decision process (MDP) [11] given that the generated molecule only depends on the molecule being modified. The MDP can be written as $M = (S, A, f, R)$ where S denotes the set of states (molecules), A denotes the set of actions (modifications), $f : S \times A \to S$ is the state transition function. For molecule modification,

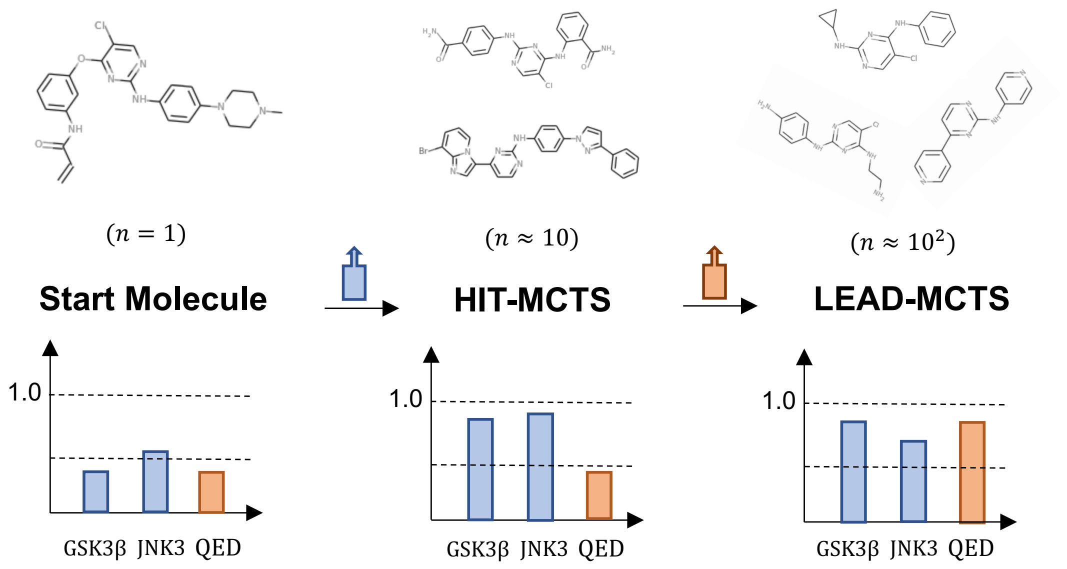


Figure 5.1: Overall framework of MolSearch. For a given start molecule, it first goes through a HIT-MCTS stage which aims to improve the biological properties, e.g., GSK3β and JNK3, followed by a LEAD-MCTS stage where non-biological properties such as QED are optimized. n refers to number of generated molecules and y-axis reflects the normalized scores.

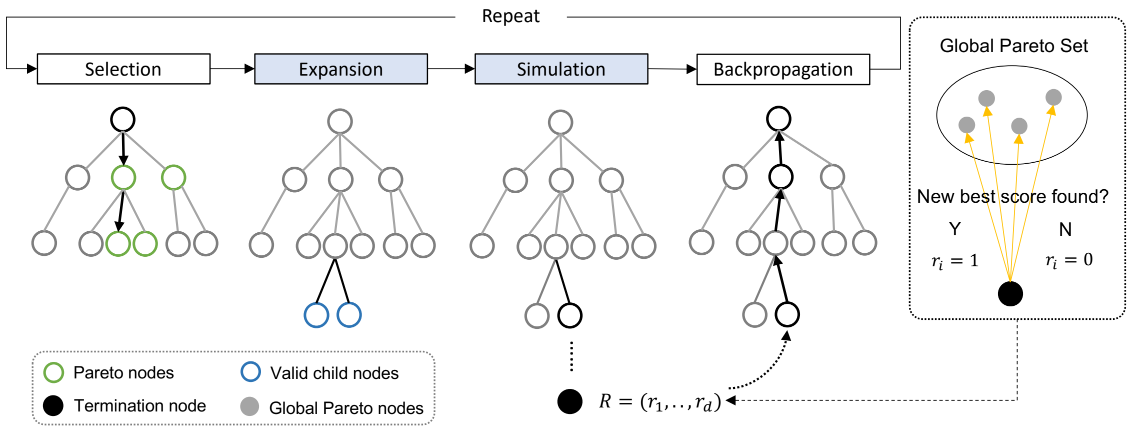


Figure 5.2: Multi-objective Monte Carlo tree search procedure. Each node represents an intermediate molecule which has a reward vector associated with it. A search iteration consists of selection, expansion, simulation, and backpropagation. For MolSearch, HIT-MCTS and LEAD-MCTS differ in the expansion and simulation policy (blue boxes).

the state transition is deterministic, i.e., $p(s_{t+1}|s_t, a_t) = 1$ for a given state-action pair. That is to say, by taking a modification action, the current molecule reaches the next molecule with that modification with probability 1. $R : S \to \mathbb{R}^d$ is the reward received for a given state, where $d > 1$ if multiple reward objectives are considered. The goal is to take the action that maximizes the expected reward, which can be approximated as Eq (5.1) under repeated simulations [34]:

$$Q(s, a) = \frac{1}{N(s, a)} \sum_{i=1}^{N(s)} \mathbb{I}_i(s, a) z_i, \tag{5.1}$$

where $N(s)$ denotes the simulation times starting from state s and $N(s, a)$ is the times that action a has been taken from state s. $\mathbb{I}_i(s, a)$ is an indicator function with value 1 if action a is selected from state s at i-th round, 0 otherwise. z_i is the final reward for i-th simulation round starting from state s. A larger value of $Q(s, a)$ indicates higher expected reward by taking action a from state s.

5.1.1 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) adopts a tree structure to perform simulations and estimate the value of actions. Meanwhile it also uses the previously estimated action values to guide the search process towards higher rewards [17]. The basic MCTS procedure consists of four steps per iteration:

a) *Selection*. Starting from the root node, a best child is recursively selected until a leaf node, i.e., a node that has not been expanded or terminated, is reached.

b) *Expansion*. The selected leaf node is expanded based on a policy until the maximum number of child nodes is reached.

c) *Simulation*. From each child node, recursively generate the next state until termination and get the final reward.

d) *Backpropagation*. The reward is backpropagated along the visited nodes to update their statistics until the root node.

The process is repeated until a certain computational budget is met. The most important step of MCTS is the *selection* step where a criterion needs to be determined to compare different child nodes. The most commonly used criteria is the upper confidence bound (UCB1) [6, 56] in which a child node is selected to maximize:

$$UCB1 = \bar{X}_j + \sqrt{\frac{2\ln n}{n_j}},$$

where $\bar{X}_j$ is the averaged reward obtained so far for node j, n_j denotes times of node j being selected and n is the total times of iteration. The first term $\bar{X}_j$ favors exploitation, i.e., choose the node with greater average performance; while the second term $\sqrt{\frac{2\ln n}{n_j}}$ votes for exploration, i.e., choose nodes that have not been visited so far. UCB1 balances between exploitation and exploration to avoid being trapped in local optimums.

For single-objective MCTS, UCB1 is a scalar and maximization picks the node with largest value. For multi-objective MCTS, the reward becomes a vector and the comparison is no longer straightforward. Next we formally define each component for multi-objective MCTS under the context of molecular generation.

5.1.2 Multi-objective Monte Carlo Tree Search

For molecular generation, each node of the tree (e.g., v_j) represents an intermediate molecule. It is associated with a molecule state s_j, number of visits n_j, and a reward vector $\mathbf{X}_j = (x_1, .., x_d) \in \mathbb{R}^d$

Algorithm 5.1: UCT algorithm for MO-MCTS.

Input: root node v_0 with state s_0, computation budge N, maximum number of child K, exploration scalar λ.

```
1:  function SEARCH($v_0$)
2:      for $i = 1, .., N$ do
3:          $v_l$ = SELECTION($v_0$)            // $v_l = v_{\text{leaf}}$
4:          $v_c$ = EXPAND($v_l$)               // $v_c = v_{\text{child}}$
5:          $\mathbf{r}_c$ = SIMULATION($v_c$)
6:          BACKPROP($v_c, \mathbf{r}_c$)
7:      return $v_0'$;
8:  function SELECTION($v$)
9:      while $v$ is fully expanded do
10:         for $k = 1, .., K$ child node do
11:             $\mathbf{U}_k = \frac{\mathbf{X}_k}{n_k} + \lambda\sqrt{\frac{4\ln n + \ln d}{2n_k}}$
12:         $V_p$ = ParetoNodeSet($\mathbf{U}_1, .., \mathbf{U}_k$)
13:     return $V_p$
```

where d is the number of objectives. Without loss of generality, we assume that each objective is to be maximized. Before presenting how the reward is calculated, we first introduce the following definitions regarding comparisons between vectors:

Definition 1. Pareto Dominate. Given two points $X = (x_1, .., x_d)$ and $X' = (x_1', .., x_d')$, X is said to *dominate* X', i.e., $X \geq X'$ if and only if $x_i \geq x_i', \forall i = 1, .., d$. X is said to *strictly dominate* X', i.e., $X > X'$ if and only if $X \geq X'$ and $\exists i$ such that $x_i > x_i'$.

Definition 2. Pareto Front. Given a set of vectors $\mathcal{A} \subset \mathbb{R}^d$, the non-dominant set $P_{\mathcal{A}}$ in $\mathcal{A}$ is defined as:

$$P_{\mathcal{A}} = \{X \in \mathcal{A} : \nexists X' \in \mathcal{A}\ s.t.\ X' > X\}$$

The *Pareto front* consists of all non-dominated points [121].

For a Monte Carlo search tree, we maintain a global pool of all the Pareto molecules found so far. At each simulation round, given a termination state (molecule) with property score $\mathbf{S} = (s_1, .., s_d) \in \mathbb{R}^d$, by comparing it with all Pareto molecules in the global pool, the reward vector $\mathbf{R} = (r_1, .., x_d) \in \mathbb{R}^d$ of this state is defined as:

$$r_i = \frac{1}{N_p}\sum_{l=1}^{N_p} \mathbb{I}[s_i > s_i^l], \qquad \forall i = 1, \ldots, d$$

where N_p is the number of Pareto molecules and s_i^l is the i-th property value of Pareto molecule l. We also update the global Pareto pool by adding new Pareto molecules if found and removing invalid ones based on the comparison result. The reward $\mathbf{R}$ will be used for backpropagation with the update formula:

$$n_v \leftarrow n_v + 1, \quad \mathbf{X}_v \leftarrow \mathbf{X}_v + \mathbf{R}, \quad v \leftarrow \text{parent of } v,$$

which concludes the backward part of MCTS.

Next we present the forward part. Starting from the root node, we recursively select the best child to proceed. To determine the best child for a given parent, we calculate the utility for each child:

$$\mathbf{U}_k = \frac{\mathbf{X}_k}{n_k} + \lambda\sqrt{\frac{4\ln n + \ln d}{2n_k}},$$

where $\mathbf{X}_k$ is the average reward obtained so far, n_k and n is the times child node k being visited and the total iterations. d is the reward dimension. Based on Definition 5.1.2 and 5.1.2, we compute the Pareto node set given statistics of all child nodes. Once the set is computed, we randomly select one child in the set to proceed. Once the selection step is done, we reach a node that has never expanded before. Then we expand the leaf node and start simulations from its children, get reward and backpropagate again. The overall MCTS procedure is illustrated in Figure 5.2 and Algorithm 1. Due to space limit, we do not present the procedure of expansion and simulation in Algorithm 1 since they are the same as classic single-objective MCTS and can be found in many places such as [17]. The key component in expansion and simulation step is the policy that used to generate the next state. In MolSearch, within each search tree, expansion and simulation share the same policy to produce actions:

$$A_v = \text{actions}(s_v),$$

for each node v given current state s_v. The possible actions are obtained using transformations we will mention in the next section. Due to the large chemical space, usually there are thousands of

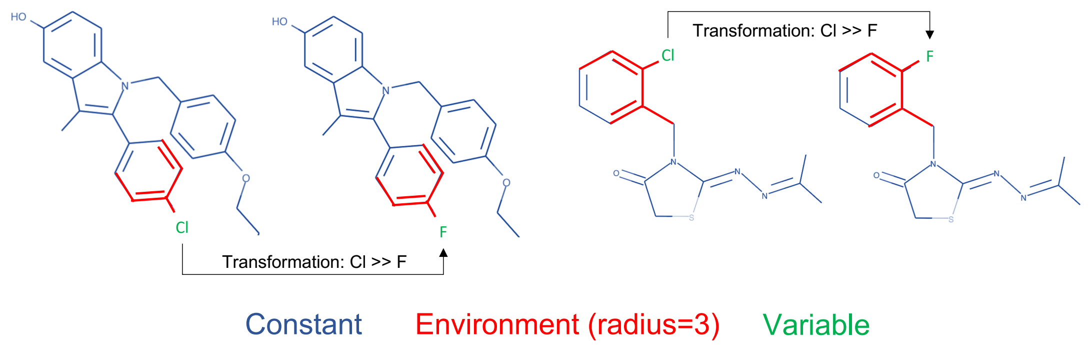


Figure 5.3: Example of design moves. A transformation is only valid conditional on the existence of certain environments.

count_stat	n	freq_stat	rule	env
# fragments	236,827	min	1	1
# environment	55,599	max	20,075	2,480
# rules	1,048,575	median	1	1
# unique rules	672,117	mean	1.78	1.56
Atom Types	C, N, O, Cl, F, P, Br, I, S			
# augment rules	436,532			
# trim rules	443,995			

Table 5.1: Statistics of rules extracted from ChEMBL on environment radius $r = 3$. # denotes "number of".

possible actions for a given state and not all of them are promising, therefore a subset of actions are selected and served as a candidate pool for both expansion and simulation.

HIT-MCTS vs LEAD-MCTS. The two search stages in MolSearch differ in how the candidates are picked given the original possible actions. In HIT-MCTS, the candidate actions are those yielding states with better property scores as compared to the current **parent** state. In LEAD-MCTS, the candidate actions are those producing states with better property scores than a **constant** threshold.

Theoretical Analysis. The theoretical analysis of multi-objective MCTS has been presented in previous work following classic concentration inequalities and union bound. Interested readers are referred to [6, 121, 22].

5.2 Design Moves

A key challenge in MolSearch is the actions to take when searching for new molecules. The modification rules should be chemically reasonable, covering a variety of modification directions, and being large in size in order to successfully navigate in the chemical space. Design moves, proposed in [7], is such an approach. It extracts transformations among molecules based on matched molecular pair (MMP) [45] and outputs a collection of rules that systematically summarize the modification of molecules that exist and chemically valid in the current large compound database such as ChEMBL [72]. The transformation rules contain both atom-wise and fragment-wise modification and for the purpose of simplicity, we refer all of them as *fragments*.

Each rule consists of three major components, a left-hand-side fragment (lhs_frag), an environment, and a right-hand-side fragment (rhs_frag), and can be written as follows:

```
lhs_frag + environment >> rhs_frag
```

An example of design move transformation is shown in Figure 5.3. Each matched molecular pair has three parts. The constant part denotes the places that remain the same before and after transformation. The variable part denotes the fragment to be replaced. The environment is the most important part in design move which characterizes the context of a transformation. The range of the context is determined by the radius r and contains all the atoms that can be reached from the fragment to be replaced within step size r. Such constraint ensures the transformation is chemically reasonable and the larger the radius r, the more likely the assumption holds true [7]. In Figure 5.3, we see that even for the same lhs_frag and rhs_frag, due to that environments are different, the transformations are treated as different transformations rules.

We summarized the statistics of all the design move rules extracted from ChEMBL based on radius $r = 3$ in Table 5.1. We see that it contains more than 1 million transformation rules with more than 600K unique pairs of fragments to be replaced. There are also more than 200K fragments and 50K environments in the total rules. For a transformation rule, the frequency it happens in the database ranges from 1 to 20K, which covers both common and rare transformations. The number of environments for the same rule also ranges from 1 to 2.5K. Given ChEMBL is one of the largest chemical databases, the rules are expected to cover all the possible moves of commonly designed molecules. Moreover, unlike most prior works which only allow atom or fragment addition, design moves contain modifications that can either increase or decrease the molecular size (436,532 vs 443,995), making it more flexible to find better modification directions.

5.2.1 Rationale of MolSearch

The last important question regarding MolSearch framework is the two-stage design in which biological properties are first optimized and then followed by non-biological properties. The reason is two-folded. First, we observe that lower non-biological property (e.g., QED and SA) values are often due to large size or large number of rings of molecules since the fragments are already chemically valid. That is to say, reducing the size of generated molecules can achieve better QED and SA scores in general. However, design move requires valid environment in order to perform modification, the larger the molecules are, the more actions could be found. Therefore, optimizing QED/SA has to come after optimizing biological properties. Second, such design is also inspired by real-world drug discovery routine that we first find drugs that are biologically active and then optimize them regarding other properties.

Another interesting property of such design is that, in general, molecules from HIT-MCTS stage are quite large, due to that HIT-MCTS modifies molecules into hits by adding property-related fragments repeatedly; However, it is fine because LEAD-MCTS will trim the molecules for a higher QED/SA score by dropping property-unrelated fragments. The entire process will ensure that the final molecules satisfies all the property requirements.

We conduct extensive experiments on benchmark tasks following [52, 128] to demonstrate the effectiveness of MolSearch. The results show that search methods can achieve comparable and sometimes superior performance compared to advanced deep learning methods given sufficient information and proper design of the algorithm.

5.3 Experiment

5.3.1 Experiment Setup

Property Objectives. We consider two biological properties that measure the inhibition of proteins related to Alzheimer disease:

- GSK3β, score of inhibiting glycogen synthase kinase-3β
- JNK3, score of inhibiting c-Jun N-terminal kinase-3

The scores are predicted probabilities of inhibition by pretrained random forest models from [52]. For non-biological properties, we follow [52, 128] and also consider drug-likeness (QED) [13] and synthesis accessibility (SA) [32] scores. The SA score (originally between [1, 10]) is reversely normalized to [0, 1]. For all scores, the higher the better. The ultimate goal is to find compounds that mostly inhibit two essential proteins in Alzheimer's such that their potency is maximized while achieving favorable medicinal chemistry properties.

Multi-objective generation setting. We consider 6 different generation settings as in [52, 128]:

- GSK3β/JNK3: inhibit GSK3β or JNK3 without constraints on QED and SA scores.
- GSK3β+JNK3: jointly inhibit GSK3β and JNK3 without constraints on QED and SA scores.
- GSK3β/JNK3+QED+SA: inhibit GSK3β or JNK3 while being druglike and easy to synthesize.
- GSK3β+JNK3+QED+SA: jointly inhibiting GSK3β and JNK3 while being druglike and easy to synthesize.

Baselines. We compare MolSearch with state-of-the-art methods from each category summarized in section 2: 1) JT-VAE [51], a method uses Bayesian optimization based on hidden representations from a VAE based on molecule fragments. 2) GCPN [133], a method uses policy gradient to finetune a pre-trained molecule generator based on GNN. 3) MolDQN [140], a method directly

Objectives	GSK3β				JNK3				GSK3β+JNK3			
Method	SR	Nov	Div	PM	SR	Nov	Div	PM	SR	Nov	Div	PM
JT-VAE	0.322	0.118	0.901	0.030	0.235	0.029	0.882	0.006	0.033	0.079	0.883	0.002
GCPN	0.424	0.116	0.904	0.040	0.323	0.044	0.884	0.013	0.035	0.080	0.874	0.002
RationaleRL	0.939	0.457	0.890	0.381	0.880	0.419	0.872	0.321	0.842	0.981	0.831	**0.686**
GA+D	0.85	1.00	0.71	0.60	0.53	0.98	0.73	0.38	0.85	1.00	0.42	0.36
MARS	1.000	0.840	0.718	0.603	0.988	0.889	0.748	**0.657**	0.995	0.753	0.691	0.518
MolDQN-emtpy	0.000	0.038	0.204	0.000	0.000	0.019	0.116	0.000	0.000	0.025	0.126	0.000
MolDQN-nonemtpy	0.341	0.304	0.856	0.089	0.175	0.288	0.857	0.043	0.050	0.421	0.858	0.018
MolSearch	1.000	0.739	0.862	**0.637** ± 0.009	1.000	0.728	0.846	0.616 ± 0.015	1.000	0.787	0.826	0.650 ± 0.009
MolSearch-5000	1.000	0.706	0.850	0.601 ± 0.023	1.000	0.685	0.845	0.579 ± 0.027	1.000	0.756	0.836	0.632 ± 0.030
Ranking				**1st**				2nd				2nd

Table 5.2: Overall performance of comparison methods on bio-activity objectives. Results of RationaleRL, MolDQN are obtained by running their open source code. Results of JT-VAE, GCPN, GA+D and MARS are taken from [52, 128]. For MolSearch, we repeat the experiments for 10 times and report the mean and standard deviation.

Objectives	GSK3β+QED+SA				JNK3+QED+SA				GSK3β+JNK3+QED+SA			
Method	SR	Nov	Div	PM	SR	Nov	Div	PM	SR	Nov	Div	PM
JT-VAE	0.096	0.958	0.680	0.063	0.218	1.000	0.600	0.131	0.054	1.000	0.277	0.015
GCPN	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
RationaleRL	0.750	0.555	0.706	0.294	0.787	0.190	0.874	0.131	0.750	0.555	0.706	0.294
GA+D	0.89	1.00	0.68	0.61	0.86	1.00	0.50	0.43	0.86	1.00	0.36	0.31
MARS	0.995	0.950	0.719	0.680	0.913	0.948	0.779	**0.674**	0.923	0.824	0.719	0.547
MolDQN-empty	0.000	0.224	0.331	0.000	0.000	0.089	0.245	0.000	0.000	0.046	0.166	0.000
MolDQN-nonempty	0.000	0.431	0.850	0.000	0.000	0.525	0.856	0.000	0.000	0.499	0.857	0.000
MolSearch	1.000	0.821	0.856	**0.702** ± 0.005	1.000	0.783	0.831	0.651 ± 0.009	1.000	0.818	0.811	**0.664** ± 0.007
MolSearch-5000	1.000	0.810	0.869	**0.704** ± 0.009	1.000	0.743	0.843	0.626 ± 0.012	1.000	0.797	0.827	**0.660** ± 0.009
Ranking				**1st**				2nd				**1st**

Table 5.3: Overall performance of comparison methods on bio-activity plus non-bioactivity objectives. Results of RationaleRL, MolDQN are obtained by running their open source code. Results of JT-VAE, GCPN, GA+D and MARS are taken from [128]. For MolSearch, we repeat the experiments for 10 times and report the mean and standard deviation.

learns the values of actions for target properties via double Q-learning and generate molecules based on that. 4) GA+D [78], a method utilizes genetic algorithm for molecule generation paired with an adversarial module to increase diversity. 5) RationaleRL [52], a method uses MCTS to find property-related fragments and then complete the graph using RL. 6) MARS [128], a method utilizes Markov sampling based on GNN and molecule fragments.

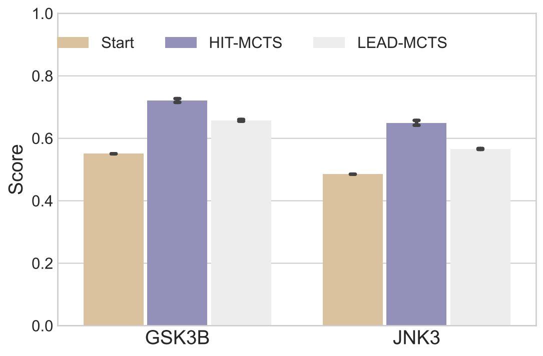
Evaluation Metrics. We evaluate the generated molecules using metrics similar to prior works [128, 52]: 1) success rate (SR): the proportion of resulted molecules that satisfy all the targeted objectives, i.e., QED ≥ 0.6, SA ≥ 0.67, GSK3$\beta \geq 0.5$, and JNK3 ≥ 0.5. 2) Novelty (Nov): the proportion of molecules that has similarity less than 0.4 compared to the nearest neighbor x_{SNN} in the reference dataset, i.e., Nov $= \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[\text{sim}(x_i, x_{\text{SNN}}) < 0.4]$ where the similarity is calculated as the Tanimoto coefficient [9] between two Morgan fingerprints [91] of molecules. The reference dataset in prior works is training data while in our work, the reference data becomes the start molecules. 3) Diversity (Div): the pair-wise dissimilarity among the generated molecules, i.e., Div $=\frac{2}{N(N-1)} \sum_{i,j=1;i\neq j}^{N}[1 - \text{sim}(x_i, x_j)]$. 4) PM: the product of SR, Nov and Div metrics, representing the possibility of generated molecules being simultaneously active, novel and diverse [128].

Start Molecules. A critical step in MolSearch is to pick the start molecules. We first download dataset from the Repurposing Hub [1], which consists of 6,758 FDA-approved and clinical trail drugs. We then cluster all the drugs based on their Tanimoto similarity using Butina algorithm [18] with threshold 0.4, a commonly used cutoff to quantify the structural similarity between molecules. It results in 5,727 small clusters, indicating that most molecules are not similar to each other. We select the centroid of each cluster, i.e., 5,727 dissimilar molecules, as the pre-processed dataset and construct start molecules from it. For benchmark objectives, to avoid making the task easier, we **remove** 1) all successful molecules, i.e., GSK3$\beta \geq 0.5$, JNK3 ≥ 0.5, QED ≥ 0.6, SA ≥ 0.67; 2) top molecules with either GSK3β or JNK3 score larger than 0.8 in the dataset. That is to say, no start molecules has biological score higher than 0.8. We then choose the remaining molecules with GSK3β and JNK3 score no less than 0.3 as the start molecules. Such selection strategy aligns with molecular optimization in reality that starts with molecules having some signals towards the desired property. There are in total 96 molecules satisfying the starting criteria.

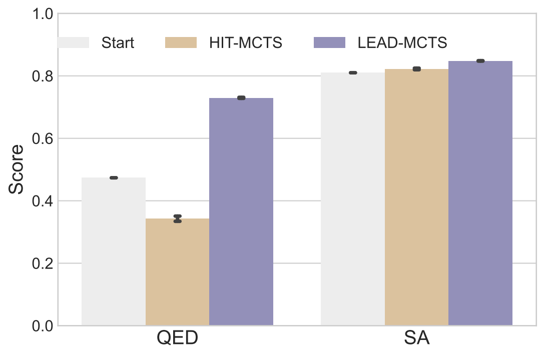
Implementation Details. For MCTS, we set the maximum level of tree depth as 5 and test different values of maximum child nodes $K = [3, 5, 7]$ and the number of simulations $N = [5, 10, 20]$. For design move, we utilize rules derived from environmental radius $r = 3$ and do not impose frequency constraint on the actions, i.e., any action with frequency ≥ 1 will be considered in each modification step. All MolSearch experiments are done on AMD EPYC CPU cores. Baselines requiring deep learning libraries are done on TITAN RTX GPUs with 24GB Memory.

Running Time. In the GSK3β+JNK3+QED+SA setting, RationaleRL takes 6 hours to finetune the model; GA+D takes 300 steps and 4 hours to reach its best performance; MARS takes 10

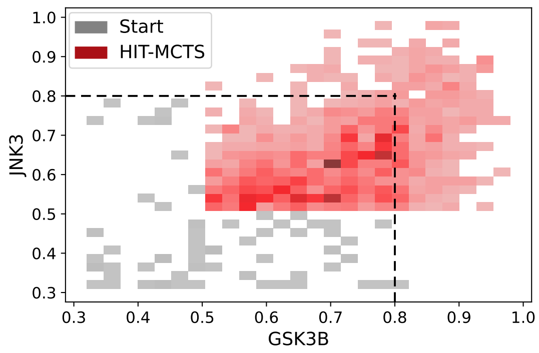
[1] https://clue.io/repurposing



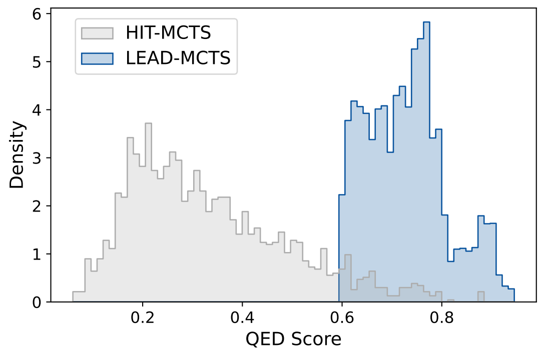
(a) biological property



(b) non-biological property



(c) biological property



(d) non-biological property

Figure 5.4: Property dynamics across MolSearch stages. (a)(b): average scores over 10 runs at each stage. (c): distribution of bioactivity scores during Start and HIT-MCTS stage. (d): QED distribution between HIT-MCTS and LEAD-MCTS stage.

hours to converge; MolDQN takes 5 and 10 hours to finish for empty and non-empty variants respectively. MolSearch takes on average 0.4-1.0 hours per molecule in both search stages (Table 5.4). Each molecule only occupies very small amount of memory and computational resources, making MolSearch much more efficient than deep learning methods regardless of computation constraints.

n_child	n_sim	Avg	Median	STD
3	10	0.4h	0.38h	0.19h
5	20	1.02h	0.87h	0.93h

Table 5.4: Running time per molecule for MolSearch.

5.3.2 Benchmark Results

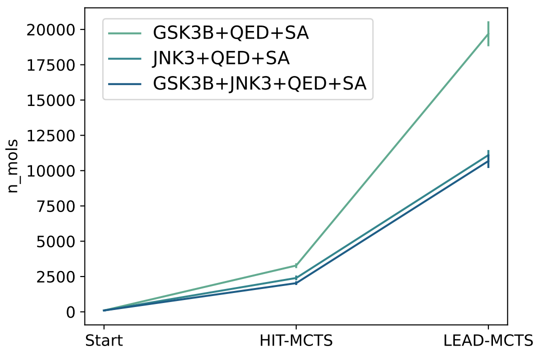
We perform the entire process of MolSearch, i.e., start molecules → HIT MCTS → LEAD MCTS for 10 times for each of the generation settings. During each search stage, we keep track of valid molecules and add them to the final set. Because the number of generated molecules is not controllable in MolSearch, we calculated the metrics for two sets of generated molecules: 1) MolSearch: all the molecules generated by MolSearch; 2) MolSearch-5000: top 5000 molecules generated by MolSearch, ranked by the average score of all properties considered in one setting, to match the number of molecules generated by other baseline methods.

Overall Performance. We summarize all the results in Table 5.2 and Table 5.3. We see that MolSearch outperforms all baselines on 3 generation settings and always rank high (1st or 2nd) in terms of PM. Specifically, for settings that considering non-bioactivity objectives, i.e., GSK3β+JNK3+QED+SA, MolSearch significantly outperforms the best baseline by 21.4% on the PM metric. Among all the metrics, MolSearch falls short on the novelty metric since it starts from known molecules and modify them into new ones. However, the novelty still ranks good via the two-stage design of MolSearch such that the generated molecules are not too similar as the original ones. The diversity of molecules generated by MolSearch always ranks high, possibly due to 1) dissimilarity of start molecules, 2) separation of different property objectives and 3) Pareto search on all objective directions.

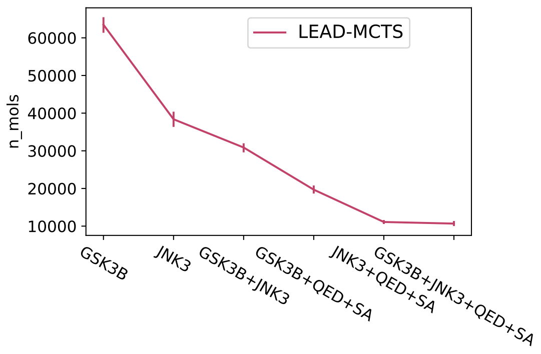
Moreover, we conduct extensive experiments for the baseline MolDQN because it is the deep learning version of MCTS that tries to learn the values of all the actions and generate molecules that maximize the values. The differences between MolDQN and MolSearch can help verify the motivation and effectiveness of MolSearch. First, MolDQN-empty starts with empty molecules and uses atom-wise actions, and the SR of generated molecules are extreme low (≈ 0.00) in all settings.

Start Molecule	GSK3β	JNK3	QED	SA
Empty	0.262	0.083	0.870	0.603
Non-empty	0.334	0.216	0.217	0.586

Table 5.5: Average scores of generated molecules by MolDQN in GSK3β+JNK3+QED+SA setting.



(a) Across stages

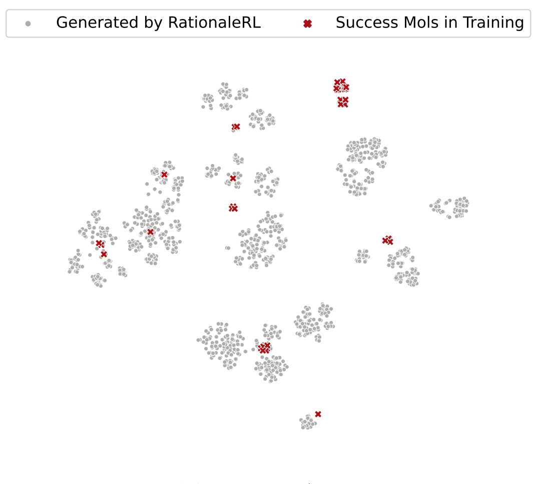


(b) Across objectives.

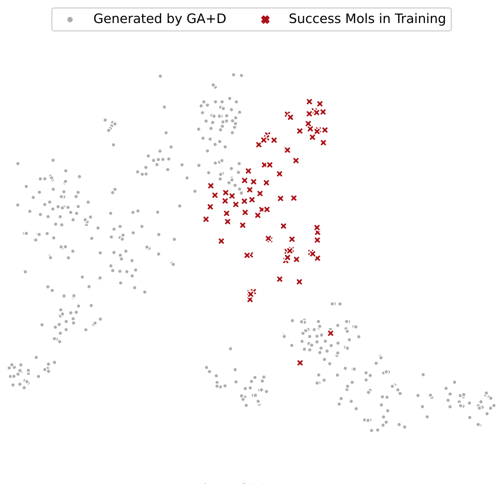
Figure 5.5: Number of generated molecules across MolSearch stages and different generation settings (10 runs).

When we look into the scores of generated molecules, as shown in Table 5.5, we find the QED and SA score of generated molecules are relatively high while GSK3β and JNK3 scores are very low. This means that QED and SA are easier to optimize than biological objectives when starting from empty molecules and using atom-wise actions. However, in most real applications, optimizing biological objectives are the major focus before one considers drug-likeness and synthesis abilities. Second, MolDQN-nonempty starts from the same molecules we used in MolSearch, however, the success rates are still low although improved compared to MolDQN-empty. This is due to that MolDQN only allows addition actions thus cannot reduce the size of molecules, making QED and SA drops significantly. Third, the low performances of both MolDQN variants imply that atom-wise actions generally works less effective compared to fragment-based actions for improving biological properties. For MolSearch, the search trees can find desired molecules with relatively small depth and width, therefore it is not necessary to use Deep Q-learning to approximate the action values. All the above observations echo the rationale of MolSearch's design.

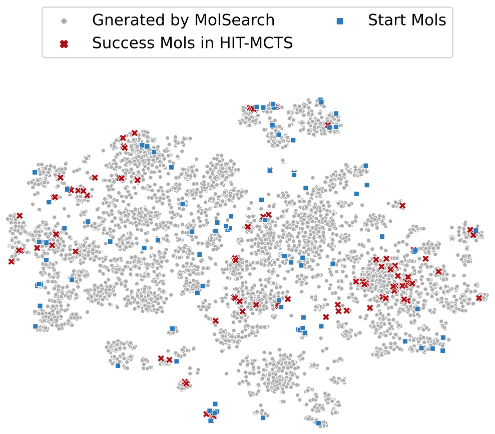
MolSearch Dynamics. We next verify whether the change of property scores across stages aligns with design motivation of MolSearch. HIT-MCTS aims to improve biological properties and Figure 5.4a confirms a significant elevation for GSK3β and JNK3 scores. LEAD-MCTS aims to improve non-biological properties and Figure 5.4b reflects such improvement especially for QED (Figure 5.4d). Figure 5.4c demonstrates that, even if we remove all successful molecules and top molecules



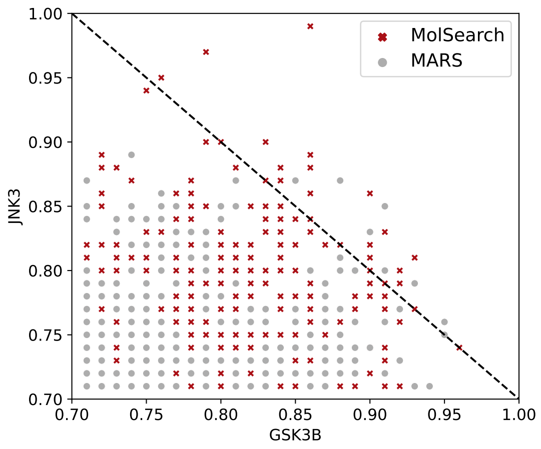
(a) RationaleRL



(b) GA+D



(c) MolSearch

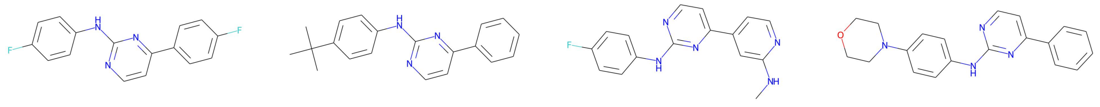


(d) MolSearch vs MARS

Figure 5.6: t-SNE visualization of generated molecules and positive molecules in the reference (training) dataset.

at start (0.3-0.8 dashed box with grey points), MolSearch is still able to find molecules with both score larger than 0.8 (red region outside dashed box), demonstrating its power. Figure 5.5a shows the number of molecules generated in each stage for three settings where both biological and non-biological objectives are considered. We observe an exponential increase from start molecules to the later two stages. GSK3β is easier to optimize as compared to JNK3. Figure 5.5b shows the number of final molecules generated by MolSearch for all settings. As the number of objectives increases, less valid molecules are found, which is reasonable.

Visualization. We compare the molecules generated under setting GSK3β+JNK3+QED+SA by



(0.91, 0.85, 0.78, 0.92) (0.93, 0.81, 0.72, 0.91) (0.89, 0.80, 0.77, 0.88) (0.80, 0.90, 0.79, 0.90)

Figure 5.7: Sample molecules generated by MolSearch in the GSK3β+JNK3+QED+SA setting with associated scores.

different methods using t-SNE plots shown in Figure 5.6 (a)-(c). The red crosses are the molecules that satisfy all the requirements in reference (training) dataset, while grey dots are molecules generated by each method. For MolSearch, there are no successful molecules in the start (reference) dataset, instead we plot the successful ones in HIT-MCTS stage. The start molecules of MolSearch are also plotted for reference (Figure 5.6c). We observe that baseline methods such as GA+D and RationaleRL generate molecules with large clusters, indicating relatively low diversity. The molecules generated by MolSearch evenly span the entire embedding space and also cover some novel regions compared to start molecules. MARS is very similar to MolSearch whose generated molecules enjoy both diversity and novelty, therefore we seek other comparison between MARS and MolSearch. As shown in Figure 5.6d, MolSearch is able to find more dominant molecules in terms of biological properties as compared to MARS (5 runs). We visualize the structure of several molecules generated by MolSearch with high property scores in Figure 5.7. Additional top ranked molecules can be found in Figure 5.9. We can see that the scaffold of highly active molecules are similar, while the non-scaffold parts are novel and enjoys a wide of range of diversity.

Figure 5.8 shows an example trajectory of MolSearch under the generation setting GSK3β + JNK3 + QED + SA. The property scores for the start molecule are relatively low. After HIT-MCTS stage, the generated molecules obtain higher GSK3β and JNK3 score by replacing certain substructures of the original molecule while also keeping certain original substructures. As we also can see, the QED score for HIT molecules are extremely low due to their large size. After LEAD-MCTS stage, the QED scores of the final molecules are elevated by dropping fragments that are less property related. The scaffold of the final molecules are not simply substructure of start molecules but rather a combination of fragments from start molecules and new fragments

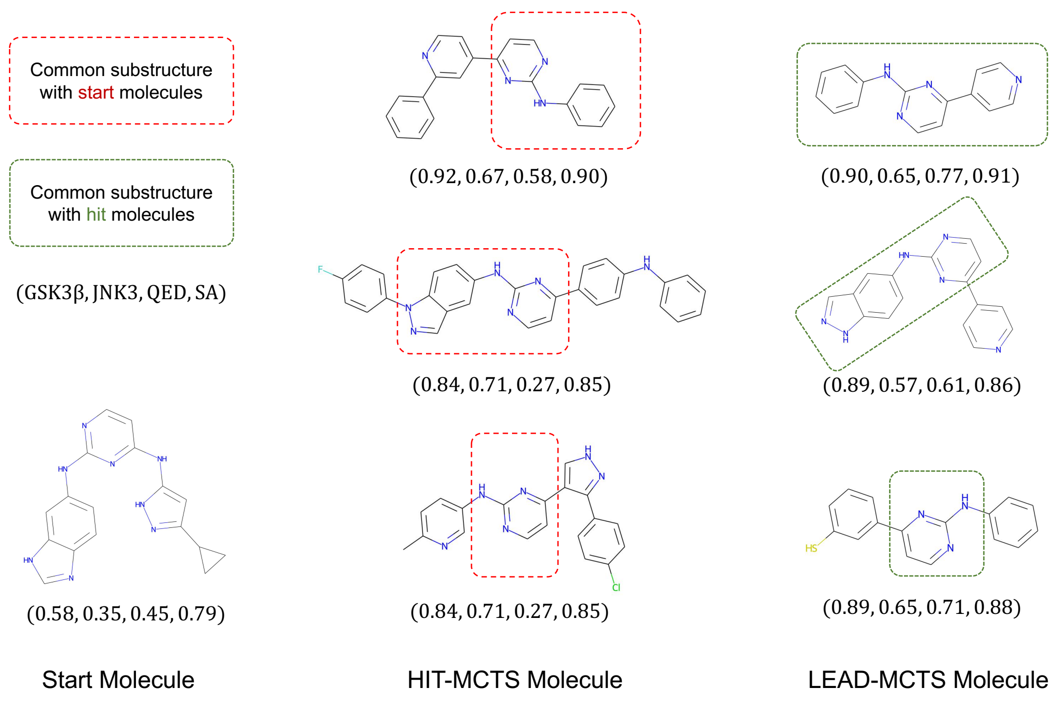


Figure 5.8: MolSearch path for generation setting GSK3β + JNK3 + QED + SA.

Setting	GSK3β+JNK3				GSK3β+JNK3+QED+SA			
K, N	SR	Nov	Div	PM	SR	Nov	Div	PM
3, 5	1.00	0.72	0.83	0.60	1.00	0.77	0.82	0.63
3, 10	1.00	0.78	0.83	0.65	1.00	0.82	0.81	0.67
3, 20	1.00	0.77	0.83	0.64	1.00	0.80	0.81	0.65
5, 5	1.00	0.76	0.83	0.63	1.00	0.79	0.82	0.65
5, 10	1.00	0.77	0.83	0.64	1.00	0.81	0.81	0.66
5, 20	**1.00**	**0.80**	**0.83**	**0.66**	1.00	0.82	0.81	0.67
7, 5	1.00	0.76	0.83	0.63	1.00	0.79	0.81	0.64
7, 10	1.00	0.78	0.83	0.65	**1.00**	**0.84**	**0.81**	**0.68**
7, 20	1.00	0.80	0.83	0.66	1.00	0.82	0.81	0.67

Table 5.6: Performance of MolSearch under different hyper-parameters for two generation settings.

from transformation rules. Also, the replacement is not completed in one round because the added fragments are relatively large, indicating the states are reached by multiple search steps instead of one.

Sensitivity Analysis We perform MolSearch under different combination of major hyper-parameters. Table 5.6 shows the overall performance of MolSearch under different combination of hyper-parameters for two generation settings. Table 5.7 shows the number of valid molecules corre-

Setting	GSK3β+JNK3			GSK3β+JNK3+QED+SA		
N/K	3	5	7	3	5	7
5	9,373	14,776	18,077	3,543	5,463	6,773
10	13,960	21,982	28,659	5,499	7,772	10,295
20	16,085	29,912	43,778	6,233	10,406	13,884

Table 5.7: Number of generated molecules by MolSearch under different hyper-parameters for two generation settings.

sponding to Table 5.6. We observe that the performance is not very different regarding different hyper-parameters, but rather the number of generated molecules are highly affected by these hyper-parameters. Because maximum number of child nodes and simulations rounds actually increases the search range such that more molecules can be found along the way.

5.3.3 Discussion

The extensive experiments of MolSearch demonstrated that given proper design and sufficient information, search-based method is also able to find molecules that satisfy multiple property requirements simultaneously with performance comparable to advanced methods using deep learning and reinforcement learning, while being much more time efficient. For MolSearch, the benefits comes from several aspects. For example, the two-stage design increases the novelty of generated molecules; Treating different objectives separately improve the diversity of the generated molecules; Fragment-based actions and starting from existing molecules maintain the synthesis abilities and drug-likeness of generated molecules.

Additional to properties in benchmark tasks, MolSearch can be easily adopted into real drug discovery projects targeting other objectives. For example, replacing GSK3β and JNK3 scoring models with COVID related predictors [53] may lead to the identification of novel and synthesizable compounds. Properties other than QED/SA, such as solubility and ADMET properties can also be included to search for more promising candidates.

MolSearch also has its own limitations. First, the bioactivity scores drop in LEAD-MCTS compared to HIT-MCTS although it is still significantly higher than start molecules (Figure 5.4a). It is because the child nodes only need to maintain bioacitivity score above 0.5 threshold in LEAD-

MCTS in exchange of higher non-bioacitvity scores. It is possible to improve the situation by setting more strict constraint during LEAD-MCTS. Second, the evaluation metrics are calculated based on unique molecules found in the search process, however, we do observe the molecules generated in LEAD-MCTS often contains many duplicates and thus causes redundancy. Third, for objectives that has relatively clear structural requirement, e.g., binding to a specific protein target, MolSearch is able to find desired molecules. However, if the objective is not sensitive to structure changes, i.e., regulation effects of multiple genes, then MolSearch, or any other related methods works less effectively. Last but not least, the scoring models are not perfect in reality since they also come form machine learning models, which may affect the quality of generation results.

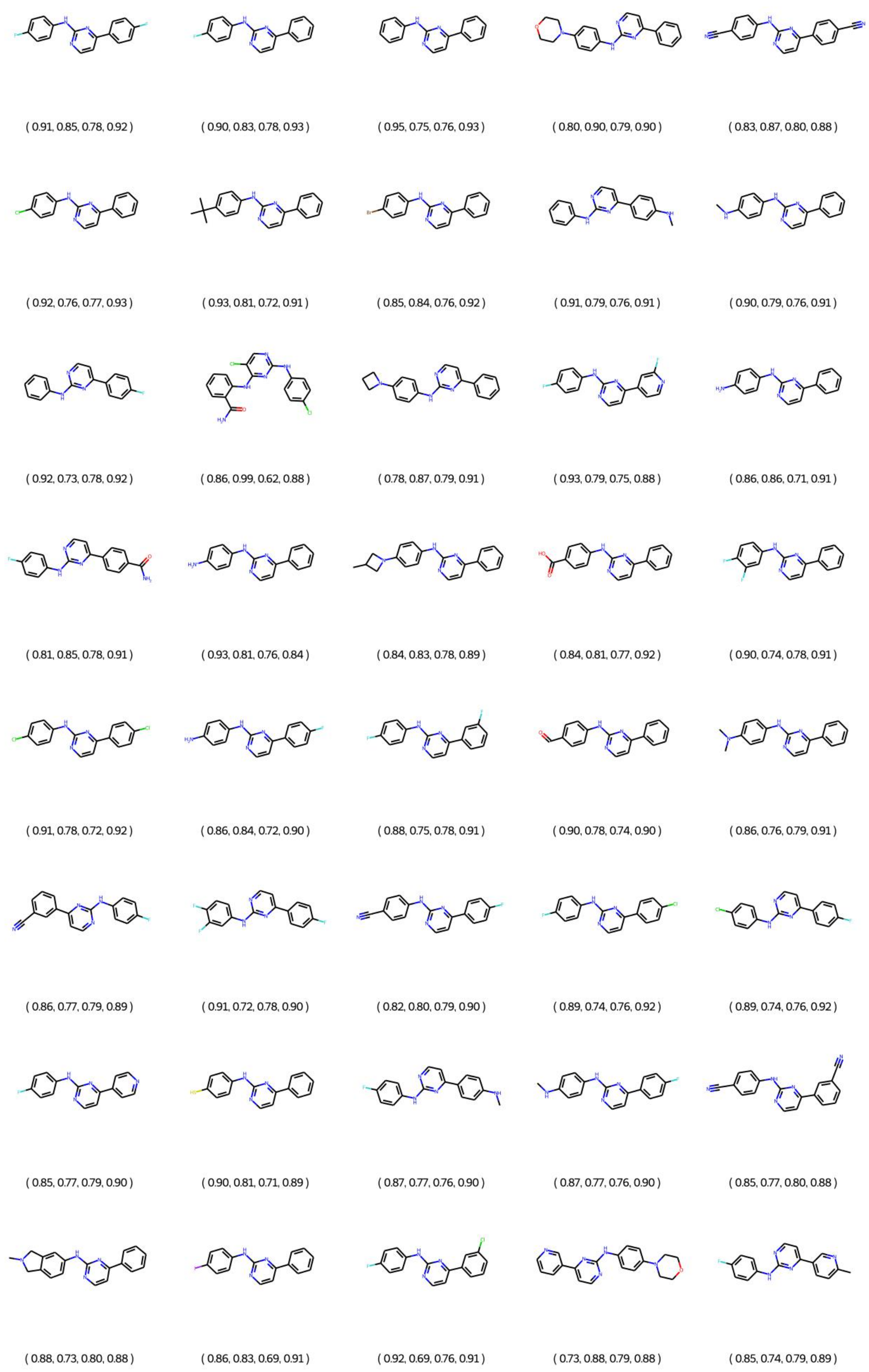


Figure 5.9: Top 40 molecules generated by MolSearch base on average score for GSK3β + JNK3 + QED + SA.

CHAPTER 6

CONCLUSION

In this dissertation, we explored and proposed novel machine learning algorithms for drug discovery from the aspects of robustness, prior knowledge and molecular design. First, we were able to better predict gene expression change, which serves as a building block in discovering therapeutic drug candidates for almost any kind of disease, via a robust learning framework. Such a framework effectively mitigates the negative effect of noisy labels due to variations in the high-throughput screening procedures. Second, by leveraging domain knowledge, we designed a novel contrastive pretraining framework for graph neural networks that can help relevant downstream tasks in the drug discovery domain. The proposed method is able to speed up training significantly under most conditions. Last, we proposed a search-based framework for molecular generation and optimization, which generates molecules with desired properties while being computationally very efficient. We demonstrated all the proposed methods through comprehensive experiments. The results indicate great potential of utilizing machine learning algorithms in improving the modern drug discovery process.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Steven Abney. Understanding the yarowsky algorithm. *Computational Linguistics*, 30(3):365–395, 2004.

[2] Sungsoo Ahn, Junsu Kim, Hankook Lee, and Jinwoo Shin. Guiding deep molecular optimization with genetic exploration. *arXiv preprint arXiv:2007.04897*, 2020.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

[4] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 233–242. JMLR. org, 2017.

[5] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019.

[6] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[7] Mahendra Awale, Jérôme Hert, Laura Guasch, Sereina Riniker, and Christian Kramer. The playbooks of medicinal chemistry design moves. *Journal of Chemical Information and Modeling*, 61(2):729–742, 2021.

[8] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. Unsupervised inductive whole-graph embedding by preserving graph proximity. 2019.

[9] Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):1–13, 2015.

[10] Tanya Barrett, Stephen E Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F Kim, Maxim Tomashevsky, Kimberly A Marshall, Katherine H Phillippy, Patti M Sherman, Michelle Holko, et al. Ncbi geo: archive for functional genomics data sets—update. *Nucleic acids research*, 41(D1):D991–D995, 2012.

[11] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957.

[12] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[13] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

[14] Benjamin E Blass. *Basic principles of drug discovery and development*. Elsevier, 2015.

[15] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

[16] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. Citeseer, 1998.

[17] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[18] Darko Butina. Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets. *Journal of Chemical Information and Computer Sciences*, 39(4):747–750, 1999.

[19] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 11–14. Springer, 2009.

[20] Bin Chen, Li Ma, Hyojung Paik, Marina Sirota, Wei Wei, Mei-Sze Chua, Samuel So, and Atul J Butte. Reversal of cancer gene expression correlates with drug efficacy and reveals therapeutic targets. *Nature communications*, 8(1):1–12, 2017.

[21] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[22] Weizhe Chen and Lantao Liu. Pareto monte carlo tree search for multi-objective informative planning. *arXiv preprint arXiv:2111.01825*, 2021.

[23] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[24] Mark Culp and George Michailidis. An iterative algorithm for extending learners to a semi-supervised setting. *Journal of Computational and Graphical Statistics*, 17(3):545–571, 2008.

[25] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, 2018.

[26] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[27] Mostafa Dehghani, Aliaksei Severyn, Sascha Rothe, and Jaap Kamps. Learning to learn from weak supervision by full supervision. *arXiv preprint arXiv:1711.11383*, 2017.

[28] Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th international conference on world wide web*, pages 1045–1052. International World Wide Web Conferences Steering Committee, 2017.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[30] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. Citeseer, 2014.

[31] Daniel C Elton, Zois Boukouvalas, Mark D Fuge, and Peter W Chung. Deep learning for molecular design—a review of the state of the art. *Molecular Systems Design & Engineering*, 4(4):828–849, 2019.

[32] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):1–11, 2009.

[33] Kaitlyn M Gayvert, Neel S Madhukar, and Olivier Elemento. A data-driven approach to predicting successes and failures of clinical trials. *Cell chemical biology*, 23(10):1294–1301, 2016.

[34] Sylvain Gelly and David Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.

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

[36] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. 2016.

[37] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[38] Phillip I Good. *Permutation, parametric, and bootstrap tests of hypotheses*. Springer Science & Business Media, 2006.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[40] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*, pages 8527–8537, 2018.

[41] Jiangfan Han, Ping Luo, and Xiaogang Wang. Deep self-learning from noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5138–5147, 2019.

[42] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.

[43] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265*, 2019.

[44] 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 International Conference on Knowledge Discovery & Data Mining*, pages 1857–1867, 2020.

[45] Jameed Hussain and Ceara Rea. Computationally efficient algorithm to identify matched molecular pairs (mmps) in large data sets. *Journal of chemical information and modeling*, 50(3):339–348, 2010.

[46] Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019.

[47] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 547–556. ACM, 2014.

[48] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086, 2014.

[49] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[50] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. *arXiv preprint arXiv:1712.05055*, 2017.

[51] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv:1802.04364*, 2018.

[52] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Multi-objective molecule generation using interpretable substructures. In *International Conference on Machine Learning*, pages 4849–4859. PMLR, 2020.

[53] Govinda B Kc, Giovanni Bocci, Srijan Verma, Md Mahmudulla Hassan, Jayme Holmes, Jeremy J Yang, Suman Sirimulla, and Tudor I Oprea. A machine learning platform to estimate anti-sars-cov-2 activities. *Nature Machine Intelligence*, 3(6):527–535, 2021.

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

[55] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[56] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[57] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[58] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079, 2016.

[59] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[60] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR, 2017.

[61] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[62] Kenneth Lange, David R Hunter, and Ilsoon Yang. Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics*, 9(1):1–20, 2000.

[63] Jisoo Lee and Sae-Young Chung. Robust training with ensemble consensus. *arXiv preprint arXiv:1910.09792*, 2019.

[64] Yong Jae Lee and Kristen Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR 2011*, pages 1721–1728. IEEE, 2011.

[65] Xiaodan Liang, Si Liu, Yunchao Wei, Luoqi Liu, Liang Lin, and Shuicheng Yan. Towards computational baby learning: A weakly-supervised approach for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 999–1007, 2015.

[66] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[67] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. *arXiv preprint arXiv:2102.01189*, 2021.

[68] Eran Malach and Shai Shalev-Shwartz. Decoupling" when to update" from" how to update". In *Advances in Neural Information Processing Systems*, pages 960–970, 2017.

[69] Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52(6):1686–1697, 2012.

[70] Hamed Masnadi-Shirazi and Nuno Vasconcelos. On the design of loss functions for classification: theory, robustness to outliers, and savageboost. In *Advances in neural information processing systems*, pages 1049–1056, 2009.

[71] Nicholas A Meanwell. Synopsis of some recent tactical application of bioisosteres in drug design. *Journal of medicinal chemistry*, 54(8):2529–2591, 2011.

[72] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, et al. Chembl: towards direct deposition of bioassay data. *Nucleic acids research*, 47(D1):D930–D940, 2019.

[73] Deyu Meng, Qian Zhao, and Lu Jiang. What objective does self-paced learning indeed optimize? *arXiv preprint arXiv:1511.06049*, 2015.

[74] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *International Conference on Machine Learning*, pages 125–134, 2015.

[75] Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2020.

[76] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[77] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. Self: Learning to filter noisy labels with self-ensembling. *arXiv preprint arXiv:1910.01842*, 2019.

[78] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alán Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. *arXiv preprint arXiv:1909.11655*, 2019.

[79] Paul A Novick, Oscar F Ortiz, Jared Poelman, Amir Y Abdulhay, and Vijay S Pande. Sweetlead: an in silico database of approved drugs, regulated chemicals, and herbal isolates for computer-aided drug discovery. *PloS one*, 8(11):e79568, 2013.

[80] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[81] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[82] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1944–1952, 2017.

[83] Zhen Peng, Yixiang Dong, Minnan Luo, Xiao-Ming Wu, and Qinghua Zheng. Self-supervised graph representation learning via global context prediction. *arXiv preprint arXiv:2003.01604*, 2020.

[84] Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

[85] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020.

[86] Anand A Rajasekar, Karthik Raman, and Balaraman Ravindran. Goal directed molecule generation using monte carlo tree search. *arXiv preprint arXiv:2010.16399*, 2020.

[87] 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`.

[88] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575, 2016.

[89] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[90] Ann M Richard, Richard S Judson, Keith A Houck, Christopher M Grulke, Patra Volarath, Inthirany Thillainadarajah, Chihae Yang, James Rathman, Matthew T Martin, John F Wambaugh, et al. Toxcast chemical landscape: paving the road to 21st century toxicology. *Chemical research in toxicology*, 29(8):1225–1251, 2016.

[91] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[92] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

[93] Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[94] Miyuki Sakai, Kazuki Nagayasu, Norihiro Shibui, Chihiro Andoh, Kaito Takayama, Hisashi Shirakawa, and Shuji Kaneko. Prediction of pharmacological activities from chemical structures with graph convolutional neural networks. *Scientific reports*, 11(1):1–14, 2021.

[95] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L Guimaraes, and Alan Aspuru-Guzik. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). 2017.

[96] Gisbert Schneider and Uli Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663, 2005.

[97] Aliaksei Severyn and Alessandro Moschitti. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 464–469, 2015.

[98] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[99] Junyuan Shang, Tengfei Ma, Cao Xiao, and Jimeng Sun. Pre-training of graph augmented transformers for medication recommendation. *arXiv preprint arXiv:1906.00346*, 2019.

[100] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

[101] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[102] Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. Computational methods in drug discovery. *Pharmacological reviews*, 66(1):334–395, 2014.

[103] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. Selfie: Refurbishing unclean samples for robust deep learning. In *International Conference on Machine Learning*, pages 5907–5915, 2019.

[104] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

[105] Aravind Subramanian, Rajiv Narayan, Steven M Corsello, David D Peck, Ted E Natoli, Xiaodong Lu, Joshua Gould, John F Davis, Andrew A Tubelli, Jacob K Asiedu, et al. A next generation connectivity map: L1000 platform and the first 1,000,000 profiles. *Cell*, 171(6):1437–1452, 2017.

[106] Govindan Subramanian, Bharath Ramsundar, Vijay Pande, and Rajiah Aldrin Denny. Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches. *Journal of chemical information and modeling*, 56(10):1936–1949, 2016.

[107] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*, 2014.

[108] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[109] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*, 2019.

[110] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5892–5899, 2020.

[111] James S Supancic and Deva Ramanan. Self-paced learning for long-term tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2379–2386, 2013.

[112] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[113] Mary M Tai. A mathematical model for the determination of total area under glucose tolerance and other metabolic curves. *Diabetes care*, 17(2):152–154, 1994.

[114] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, 2019.

[115] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5596–5605, 2017.

[116] Florin Vaida. Parameter convergence for em and mm algorithms. *Statistica Sinica*, 15(3):831, 2005.

[117] Brendan Van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In *Advances in Neural Information Processing Systems*, pages 10–18, 2015.

[118] Paroma Varma, Bryan He, Dan Iter, Peng Xu, Rose Yu, Christopher De Sa, and Christopher Ré. Socratic learning: Correcting misspecified generative models using discriminative models. *arXiv preprint arXiv:1610.08123*, 2017.

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

[120] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR (Poster)*, 2019.

[121] Weijia Wang and Michele Sebag. Multi-objective monte-carlo tree search. In *Asian conference on machine learning*, pages 507–522. PMLR, 2012.

[122] Xiaobo Wang, Shuo Wang, Jun Wang, Hailin Shi, and Tao Mei. Co-mining: Deep face recognition with noisy labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9358–9367, 2019.

[123] John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113, 2013.

[124] Godwin Woo, Michael Fernandez, Michael Hsing, Nathan A Lack, Ayse Derya Cavga, and Artem Cherkasov. Deepcop: deep learning-based approach to predict gene regulating effects of small molecules. *Bioinformatics*, 2019.

[125] Jung Hoon Woo, Yishai Shimoni, Wan Seok Yang, Prem Subramaniam, Archana Iyer, Paola Nicoletti, María Rodríguez Martínez, Gonzalo López, Michela Mattioli, Ronald Realubit, et al. Elucidating compound mechanism of action by network perturbation analysis. *Cell*, 162(2):441–451, 2015.

[126] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[127] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

[128] Yutong Xie, Chence Shi, Hao Zhou, Yuwei Yang, Weinan Zhang, Yong Yu, and Lei Li. Mars: Markov molecular sampling for multi-objective drug discovery. *arXiv preprint arXiv:2103.10432*, 2021.

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

[130] Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama, and Koji Tsuda. Chemts: an efficient python library for de novo molecular generation. *Science and technology of advanced materials*, 18(1):972–976, 2017.

[131] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[132] Michihiro Yasunaga and Percy Liang. Graph-based, self-supervised program repair from diagnostic feedback. In *International Conference on Machine Learning*, pages 10799–10808. PMLR, 2020.

[133] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.

[134] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33, 2020.

[135] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? In *International Conference on Machine Learning*, pages 10871–10880. PMLR, 2020.

[136] Wenbo Yu and Alexander D MacKerell. Computer-aided drug design methods. In *Antibiotics*, pages 85–106. Springer, 2017.

[137] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W Tsang, and Masashi Sugiyama. How does disagreement benefit co-teaching? *arXiv preprint arXiv:1901.04215*, 2019.

[138] Dingwen Zhang, Deyu Meng, and Junwei Han. Co-saliency detection via a self-paced multiple-instance learning framework. *IEEE transactions on pattern analysis and machine intelligence*, 39(5):865–878, 2016.

[139] Yi Zhang. An in-depth summary of recent artificial intelligence applications in drug design. *arXiv preprint arXiv:2110.05478*, 2021.

[140] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.

[141] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. *arXiv preprint arXiv:2010.14945*, 2020.

[142] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.