DEEP LEARNING FOR MOLECULAR PROPERTY PREDICTION

by

HEHUAN MA

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2023

To my dear family,

for all their endless trust, continuous support, and unconditional love.

ACKNOWLEDGEMENTS

Time flies. It feels like just yesterday when I started my Ph.D. in 2018. I have learned so much during this journey, and I could never have completed it without all the wonderful people I have had the privilege to meet.

I would like to sincerely thank my supervising professor, Dr. Junzhou Huang, who has given me this opportunity to pursue advanced research. Dr. Huang constantly encourages me to excel and improve myself, especially when I feel stuck in my research. His guidance has been pivotal in the completion of this dissertation. I have gained a wealth of knowledge from Dr. Huang. His extensive expertise, innovative research perspectives, and constructive advice have not only enriched my research journey but have also shaped my broader outlook on academia and life. I have learned how to immerse myself in research, effectively present my work, communicate with colleagues, engage with the academic community, and much more. All these insights will continue to motivate me in my future career pursuits.

I would like to extend my genuine gratitude to Dr. Chengkai Li, Dr. Jia Rao, and Dr. Dajiang Zhu for serving on my committee. Their invaluable suggestions and feedback have greatly enhanced my academic journey. Their expertise across diverse domains, including natural language processing, distributed systems, and medical image learning, has provided me with valuable cross-disciplinary insights. It is truly my privilege to have each of them serve on my committees and engage with my research.

I deeply appreciate the guidance from Yu Rong, Yatao Bian, Wenbing Huang, Tingyang Xu, Long-Kai Huang, Weiyang Xie, Geyan Ye, and Jiaxiang Wu during my internship in the industry. My special thanks go to Dr. Rong, who not only served as my mentor but also acted as a role model, exemplifying the qualities of a passionate researcher. I admire his enthusiasm and rigor towards research, which has also deeply impacted me, inspiring me to embody these qualities in my research endeavors.

For the past five years, it has been my pleasure to work with lab members from Scalable Modeling and Imaging and Learning Lab (SMILE). I would like to express my appreciation to Zheng Xu, Sheng Wang, and Chaochao Yan, who were the first individuals I met before joining the lab. I want to thank Zheng for being my initial mentor at SMILE. I want to thank Sheng, for his considerate and patient guidance in coding and early-stage Ph.D. research, which are invaluable to me. I reserve special thanks to my labmates and roommates, Yuzhi Guo and Xinliang Zhu, for their consistent support in both my studies and life. And a shoutout to my other roommate, Haoran Li, whose positive spirit during the pandemic is truly uplifting.

I would also like to express my gratitude to the kind staff members of the CSE department, particularly Ms. Sherri Gotcher and Ms. Ginger Dickens, for their warm assistance and support during my study at UTA.

I would like to thank my dear friend Dr. Rui Liu. We attended the same university for both our bachelor's and master's studies. She encourages me to pursue further studies in the U.S. and provides invaluable advice and support for both my academic and personal journeys.

Lastly, my deepest gratitude is dedicated to my beloved family, for their endless trust, continuous support, and unconditional love throughout my entire life.

August, 11

ABSTRACT

DEEP LEARNING FOR MOLECULAR PROPERTY PREDICTION

Hehuan Ma, Ph.D.

The University of Texas at Arlington, 2023

Supervising Professor: Junzhou Huang

Drug discovery has always been a crucial task for society, and molecular property prediction is one of the fundamental problem. It is responsible for identifying the target properties or severe side-effects, so that certain molecules can be selected as the candidates of drugs. Traditional methods usually conduct a series of biochemical experiments to test the molecular properties, which may take up to decades. Nowadays, this process can be facilitated due to the rapid growth of deep learning methods. I present my work toward solving this critical problem by utilizing deep learning techniques. My research study can be summarized in three directions: 1) designing informative and powerful molecular representation learning models; 2) exploring the imbalanced data distribution and developing corresponding techniques to further improve the prediction performance, and 3) taking advantage of the unlabeled data to address the limited labeled data problem in molecular property prediction.

How to accurately and properly represent the molecule is a dominant perspective to target molecular property prediction. Precisely depicting molecules is a crucial factor that significantly impacts property predictions. To tackle this challenge, I propose a cross-dependent graph neural network to learn and generate informative molecular representation by exploring the molecular graph structure, which takes both atom-oriented graph structure and edge-oriented graph into consideration. Moreover, the molecular data could be imbalanced since some molecules may be easily predicted while some others are not. Therefore, I explore the data distribution and propose an attentive loss function to allow the network to learn the sample importance with respect to different molecules, which further improves the model performance. Lastly, acquiring accurate molecular property information remains a demanding task due to the labor and resource-intensive nature of labeling. To address this issue, effectively utilizing unlabeled data stands out as a potential solution. I propose a robust self-training strategy to include unlabeled data to promote molecular property prediction. Furthermore, I propose a data-augmentation strategy using graph neural network by incorporating these two methods to solve a realistic problem, drug-induced liver injury (DILI) prediction, and have obtained significant improvement on this extremely small dataset, which only contains hundreds of molecules. Moreover, I a sequence-based multi-label learning method to improve the performance of the property prediction with limited data in a semi-supervised manner.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Molecular property prediction is a fundamental but crucial task of drug discovery since it helps to determine the function of new drugs. The traditional way to identify molecular property is to conduct professional experiments based on the domain knowledge of chemical experts [3, 4]. It typically starts with target identification, which requires applying data mining techniques to identify and select the function of a potential biochemical target when it is relevant to a disease condition [5, 6]. Probably hundreds of thousands of drug candidates need to go through a precise functional screening process to constitute their interactions with the drug target, which is extremely time-consuming and computationally costly. Once a drug candidate appears capable, as well as the target is identified retrospectively, a corresponding library of similar compounds is then synthesized and screened to find a drug that binds to the target with the desired effect. Thus, the discovery of an FDA-approved drug could take money and up to 12 years [3, 7]. How to promote the process is the key challenge.

Nowadays, deep learning techniques are widely known for their capability of taking advantage of huge amounts of data, which has achieved substantial success in many domains, such as computer vision [8–14], medical image processing [15–17], nature language processing [18, 19], health care [20–23], bioinformatics [24–31], etc. Meanwhile, in recent years, the amount of available compounds and biological activity data increased exponentially due to experimental techniques such as High-throughput screening (HTS) and parallel synthesis [32, 33]. Thus, deploying deep learning models to address molecular property prediction problem is promising. Although many studies have been done by utilizing various deep learning models [34–36, 36–39] over this problem, there still remain several obstacles: 1) an informative molecular representation is needed to accurately represent the molecule for further property prediction; 2) the underlying data distribution is rarely explored, which might be imbalanced; and 3) labeled property data is difficult to obtain since it requires a series of biochemical experiments or lots of computational resources to test the target property.

In this thesis, I tackle these obstacles by proposing 1) a novel GNN-based framework for generating informative molecular representation by sufficiently learning the molecular graph structure as well as the chemical features; 2) an attentive loss function to explore the data distribution and learn associated importance for each data sample; 3) a robust self-training paradigm that utilize both labeled and unlabeled data to promote the prediction performance; 4) a data augmentation strategy which takes GNN-based models as the backbone to boost the prediction of an extremely small but important molecular property, drug-induced liver injury (DILI), and 5) a sequence-based deep multi-label learning over limited data, which takes advantages of both labeled data and unlabeled in a semi-supervised manner, and improves the prediction performance for those properties with limited data. These works have been published [40–44] in several conferences and journals. In this thesis, I will present how each of them solves molecular property prediction problem as follows:

Chapter introduces a novel GNN based method for generating informative molecular representation by employing multi-view modeling with a cross-dependent message passing scheme to improve information sharing across diverse perspectives. This framework significantly outperforms existing models on challenging benchmarks, demonstrating enhanced accuracy and interpretability.

In order to explore the underlying data distribution, **Chapter 2.6** proposes a self-attention mechanism that assigns learnable weights to each data sample based on its gradient norm, acknowledging that samples should not be treated equally during the network training due to the easy/hard sample imbalanced problem. Specifically, an attentive loss function is designed to learn the associated weights for each data sample automatically.

Semi-supervised learning is a widely used paradigm to assist supervised learning problem. In **Chapter 3.5**, a robust self-training strategy is developed to involve more unlabeled data to promote the molecular property prediction tasks. A teacher-student paradigm is employed, and robust loss function is explored to handle the label noise. Moreover, **Chapter 4.4** addresses a real-world challenging DILI (Drug-Induced Liver Injury) prediction task, which is crucial for drug discovery due to its potential toxicity to humans. DILI is considered an extremely small dataset with only a few hundred labeled molecules. We propose a property augmentation strategy to integrate extensive training data with additional property information. Empirical tests showcase our method's superiority over existing baselines. Last, **Chapter 5.5** presents a multi-label learning method that improves the prediction performance for limited data by utilizing a sequence-based model as the backbone model, which demonstrates that multi-label learning is an effective approach to improve the performance for limited data.

Chapter 6.5 briefly summarizes these research studies, and discusses several future works that towards my goal for the next step.

CHAPTER 2

CROSS-DEPENDENT GRAPH NEURAL NETWORKS FOR MOLECULAR PROPERTY PREDICTION

As mentioned before, the crux of molecular property prediction is to generate meaningful representations of the molecules. One promising route is to exploit the molecular graph structure through Graph Neural Networks (GNNs). Both atoms and bonds significantly affect the chemical properties of a molecule, so an expressive model ought to exploit both node (atom) and edge (bond) information simultaneously. Inspired by this observation, we explore multi-view modeling with graph neural network (MVGNN) to enable more accurate predictions of molecular properties. To further enhance the expressive power of MVGNN, we propose a cross-dependent message-passing scheme to enhance information communication of different views. The overall framework is termed CD-MVGNN. Lastly, we theoretically justify the expressiveness of the proposed model in terms of distinguishing non-isomorphism graphs. Extensive experiments demonstrate that CD-MVGNN achieves remarkably superior performance over state-of-the-art models on a variety of challenging benchmarks. Meanwhile, visualization results of the node importance are consistent with prior knowledge, which confirms the interpretability power of CD-MVGNN.

2.1 Introduction

Molecular property prediction is a fundamental but challenging task in drug discovery, and attracts increasing attention in the last decades. For example, designing molecular fingerprints based on the radial group of the molecular structure for property prediction [36]. However, traditional molecular property prediction methods usually i) require chemical experts to conduct professional experiments to validate the property label, ii) desire high R&D cost and massive amount of time, and iii) ask for specialized models for different properties, which lacks generalization capacity [3, 45].

To date, Graph Neural Networks (GNNs) have gained more popularity due to its capability of modeling graph structured data. Successes have been achieved in various domains, such as social network [46, 47], knowledge-graphs [48, 49], and recommendation systems [50, 51]. Molecular property prediction is also a promising application of GNNs since a molecule could be represented as a topological graph by treating atoms as nodes, and chemical bonds as edges. Compared with other representations for molecules, such as SMILES [52], which represents molecules as sequences but loses structural information, graph representation of the molecules can naturally capture the information from the molecular structure, including both the nodes (atoms) and edges (bonds). In this sense, a molecular property prediction task is equivalent to a supervised graph classification problem (see, for example, toxicity prediction [53] and protein interface prediction [54]).

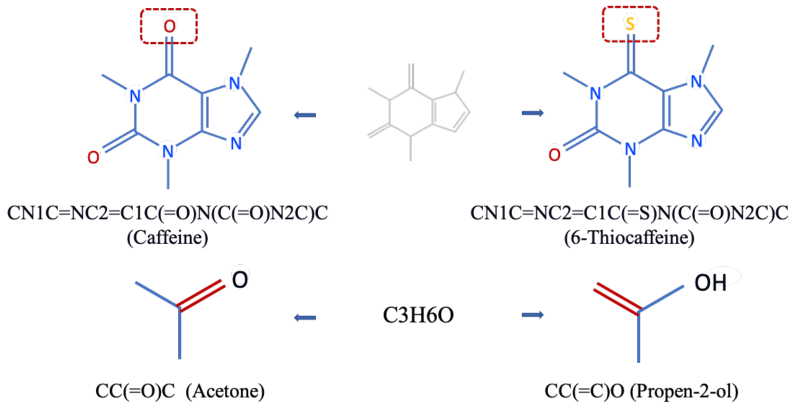


Figure 2.1: The upper two molecules share the same bond structures, but contain different atoms. The lower two molecules share the same atoms, but equip with different bonds.

Despite the fruitful results obtained by GNNs, there remains three limitations: 1) Most of the GNN models focus either on the embedding of nodes or edges. However, in many practical scenarios, nodes and edges play equally important roles. Specifically, molecules with different atoms (nodes) but same bonds (edges) are distinct compounds with different properties, and so as to different bonds (edges) but same atoms (nodes). As shown in Figure 2.1(upper), equipped with the same bonds, only one-atom difference make the two molecules distinct Octanol/Water Partition Coefficients. Caffine is more hydrophilic while 6-Thiocaffeine is more lipophilic [55]. Similarly, in Figure 2.1(lower), the molecular formulas of Acetone and Propen-2-ol are exactly the same, but the bond difference makes Acetone behave mild irritation to human eyes, nose, skin, etc [33]. Accordingly, both nodes and edges are fairly essential for molecular property prediction. Therefore, how to properly integrate *both node and edge information in a unified manner* is the first challenge. 2) As proved in [56], the GNN model with message passing scheme is at most as strong as the WL graph isomorphism test [57], which limits the expressive power of GNNs and harms the performance of down-steam tasks. 3) Existing GNNs usually lack interpretability power, which is actually crucial for drug discovery tasks. Take molecular property prediction as an example, being aware of how the model validate the property will help practitioners figure out the key components that determine certain properties [58].

In pursuit of tackling the above challenges, we explore the idea of multi-view learning [59] and a new form of GNN: MVGNN, which contains two sub-modules that generate the graph embeddings from node and edge, respectively. Therefore, it investigates the molecular graph from two views simultaneously. Furthermore, we design a cross-dependent message passing scheme to break the expressive power barrier of MVGNN and propose CD-MVGNN. We theoretically justify that the expressiveness of CD-MVGNN is strictly more powerful than Graph Isomorphisom Network [56] in terms of distinguishing non-isomorphism graphs. Lastly, we employ a shared self-attentive aggregation module to produce the graph-level embeddings and interpretability results, as well as a disagreement loss to stabilize the training process of the multi-view pipeline. Comprehensive experiments on 11 benchmarks demonstrate the superiority of proposed CD-MVGNN model. Namely, the overall performance of CD-MVGNN and MVGNN achieve up to 3.62% improvement on classification benchmarks and 23.55% improvement on regression benchmarks compared with SOTA methods. Moreover, case studies on toxicity prediction demonstrate the interpretability power of proposed model.

2.2 Related Work

The most crucial part of addressing molecular property prediction problem is to get an accurate vector representation of the molecules. Relevant studies can be categorized into three aspects: hand-crafted molecular fingerprints based methods, SMILES sequence based techniques, and graph structure based techniques.

Hand-crafted molecular fingerprints based methods. The traditional feature extraction method enlists experts to design molecular fingerprints manually, based on biological experiments and chemical knowledge [60], such as the property of molecular sub-structures. These types of fingerprint methods generally work well for particular tasks but lack universality. One representative approach is called circular fingerprints [36]. Circular fingerprints employ a fixed hash function to extract each layer's features of a molecule based on the concatenated features of the neighborhood in the previous layer. Extended-Connectivity Fingerprint (ECFP) [61] is one of the most famous examples of hash-based fingerprints. The generated fingerprint representations usually go through machine learning models to perform further predictions, such as Logistic Regression [62], Random Forest [63], and Influence Relevance Voting

(IRV) [64]. Nonetheless, this type of hand-crafted fingerprint has a notable problem: since the characteristic of the hash function is non-invertible, it might not be able to catch enough information when being converted.

SMILES sequence based techniques. SMILES sequence based models, such as Seq2seq Fingerprint [38], spot the potentially useful information of the molecular SMILES sequence data by adequately training them using Recurrent Neural Networks (RNNs), in order to obtain the vector representation of the molecule. These vectors then go through other supervised models to perform property prediction, e.g., GradientBoost [65]. The SMILES-based models are inspired by the sequence learning in Natural Language Processing [18], which takes an unlabeled dataset as the input to convert a SMILES to a fingerprint, then recovers the fingerprint back to a sequence representation for better learning.

Graph structure based techniques. A molecule could be represented as a graph based on its chemical structure, e.g., consider the atoms as the nodes, and the chemical bonds between the atoms as the edges. Thus, many graph theoretic algorithms could be applied to represent a molecule by embedding the graph features into a continuous vector [35,66]. A noted study proposed the idea of neural fingerprints, which applies convolutional neural networks on graphs directly [34]. The difference between neural fingerprints and circular fingerprints is the replacement of the hash function. Neural fingerprints apply a non-linear activated densely connected layer to generate the fingerprints. These kind of deep Graph Convolutional Neural Networks are established by learning a function on the graph node features and the graph structure matrix representation [35, 54]. Other graph-based models such as the Weave model have also been proposed [67]. The Weave model is another graph-based convolutional model. The key difference between the Weave model and neural fingerprints [34] is the updating procedure of the atom features. It combines all the atoms in a molecule with their matching pairs instead of the neighbors of the atoms. More relevant research that focus on exploiting the molecular graphs with graph convolutional network have been studied recently, e.g., [68, 69] have involved the 3D information of the molecules to help exploit the molecular graph structure. Other attempts such as [70, 71] turn to develop aggregation weights learning schemes based on the prior knowledge of Graph Attention Network [46], while [72] propose to learn different weights for different types of edges. Moreover, [73] proposes a framework to implement message passing process between each atom to form a molecular representation. Inspired by this work, [1, 74] convert the message passing process to bond-wise instead of atom-wise, and [75] explores more information by considering both atom messages and the corresponding incoming bond messages. [76] introduces multilevel graph structures based on the interactions between atom-pairs.

2.3 Preliminaries on Molecular Representations and Generalized GNNs

We abstract a molecule c as a topological graph $G_c = (\mathcal{V}, \mathcal{E})$, where $|\mathcal{V}| = p$ refers to the set of p nodes (atoms) and $|\mathcal{E}| = q$ refers to a set of q edges (bonds). $\mathcal{N}_v$ denotes the neighborhood set of node v. We denote the feature of node v as $\mathbf{x}_v \in \mathbb{R}^{d_n}$ and the feature of edge (v, k) as $\mathbf{e}_{vk} \in \mathbb{R}^{d_e}$[1]. d_n and d_e refer to the feature dimensions of nodes and edges, respectively. Exemplar node and edge features are the chemical relevant features such as atomic mass and bond type. Please refer to Section 2.4.2 for detailed feature extraction process. Properties of a molecule $\mathbf{y}$ constitute the targets of the predictive task. Given a molecule c and its associated graph representation G_c, molecular property prediction aims to predict the properties $\mathbf{y}_c$ according to the embedding ξ_c of G_c. The values of $\mathbf{y}$ are either categorical values (e.g., toxicity and

[1]With a bit abuse of notations, $\mathbf{e}_{vk}$ can represent either the edge (v, k) or the edge features.

permeability [77, 78]) for classification tasks or real values (e.g., atomization energy and the electronic spectra [79]) for regression tasks.

Generalized GNNs. Most of the GNN models are built upon the message passing process, which aggregates and passes the feature information of corresponding neighboring nodes to produce new hidden states of the nodes. After the message passing process, all hidden states of the nodes are fed into a readout component, to produce the final graph-level embedding. Here we present a generalized version of the message passing scheme. Suppose there are L iterations/layers, and iteration l contains K_l hops. In iteration l, the k-th hop of message passing can be formulated as,

$$\begin{aligned} \mathbf{m}_v^{(l,k)} &= \mathrm{AGG}^{(l)}(\{\mathbf{h}_v^{(l,k-1)}, \mathbf{h}_u^{(l,k-1)}, \mathbf{e}_{uv} \mid u \in \mathcal{N}_v\}), \qquad (2.1) \\ \mathbf{h}_v^{(l,k)} &= \mathrm{MLP}^{(l)}(\mathbf{m}_v^{(l,k)}), \end{aligned}$$

where we make the convention that $\mathbf{h}_v^{(l,0)} := \mathbf{h}_v^{(l-1,K_{l-1})}$. $\mathrm{AGG}^{(l)}$ denotes the aggregation function, $\mathbf{m}_v^{(l,k)}$ is the aggregated message, and $\mathrm{MLP}^{(l)}$ is a multi-layer perceptron[2]. There are several popular choices for the aggregation function $\mathrm{AGG}^{(l)}$, such as mean, max pooling and the graph attention mechanism [46]. Note that for one iteration of message passing, there are a layer of trainable parameters (parameters inside $\mathrm{AGG}^{(l)}$ and $\mathrm{MLP}^{(l)}$). These parameters are shared across the K_l hops within iteration l. After L iterations of message passing, the hidden states of the last hop in the last iteration are used as the embeddings of the nodes, i.e., $\mathbf{h}_v^{(L,K_L)}, v \in \mathcal{V}$. Lastly, a READOUT operation is applied to generate the graph level representation,

$$\mathbf{h}_G = \mathrm{READOUT}(\{\mathbf{h}_v^{(0,K_0)}, ..., \mathbf{h}_v^{(L,K_L)} \mid v \in \mathcal{V}\}). \qquad (2.2)$$

If choosing the sum aggregation with a learnable parameter $\epsilon^{(l)}$, i.e.,
$\mathrm{AGG}^{(l)}(\{\mathbf{h}_v^{(l,k-1)}, \mathbf{h}_u^{(l,k-1)}, \mathbf{e}_{uv} | u \in \mathcal{N}_v\}) = ((1+\epsilon^{(l)})\mathbf{h}_v^{(l,k-1)} + \sum_{u\in\mathcal{N}_v} \mathbf{h}_u^{(l,k-1)}) || (\sum_{u\in\mathcal{N}_v} \mathbf{e}_{uv})$

[2]For instance, it could be a one layer neural net, then the state update becomes $\mathbf{h}_v^{(l,k)} = \sigma(\mathbf{W}^{(l)}\mathbf{m}_v^{(l,k)} + \mathbf{b}^{(l)})$, where σ stands for the activation function.

(|| is the concatenation operation), then generalized GNN recovers graph isomorphism network (GIN) architecture [56], which provably generalizes the WL graph isomorphism test [57].

2.4 Cross-Dependent Multi-View GNN (CD-MVGNN)

In this section, we will first introduce the high-level framework of CD-MVGNN model, then illustrate each components in detail. Lastly we theoretically verify the expressive power of CD-MVGNN and the basic MVGNN architecture.

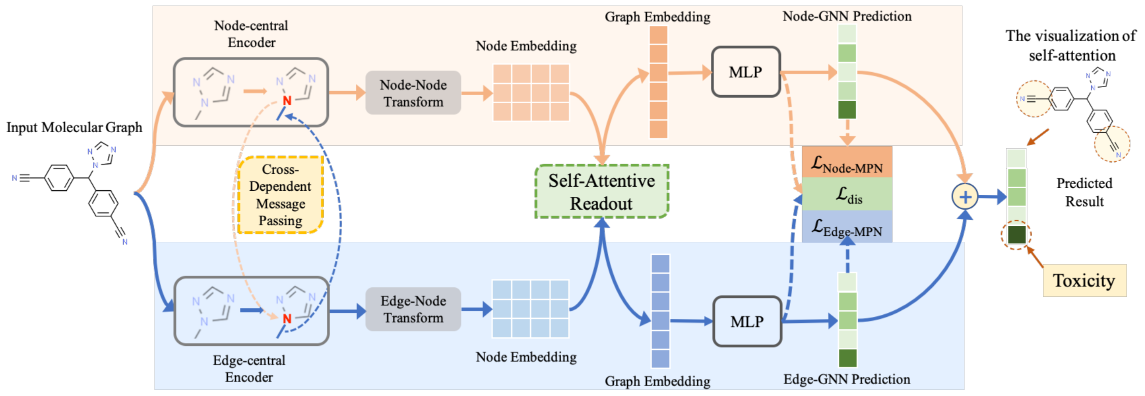
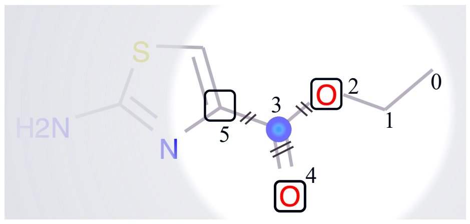


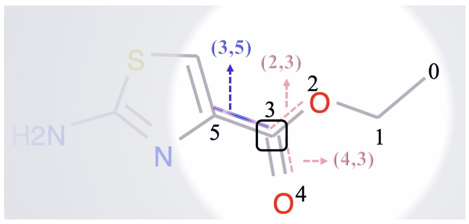
Figure 2.2: Overview of CD-MVGNN model. CD-MVGNN model passes the graph through two encoders to generate two sets of node embeddings. A cross-dependent message passing scheme is applied between two encoders to ensure the information flow circulation. A shared self-attention readout learns the node importance and produce two graph embeddings accordingly. The embeddings are then fed into two MLPs to make predictions. The final prediction is the ensemble of the two predictions. Furthermore, by visualizing the learned attentions over nodes, one can identify the atoms/functional groups that are responsible for the predictions. For example, CD-MVGNN finds out that the cyano groups contribute to the toxicity significantly.

2.4.1 Overview of CD-MVGNN

CD-MVGNN is implemented in a multi-view fashion, namely, it equally considers both atom features and bond features for constituting a molecular representation. As shown in Figure 2.2, the proposed architecture contains two concurrent sub-modules, *Node-central encoder* and *Edge-central encoder*. A *cross-dependent message passing* scheme is applied between the two encoders to enable messages circulate and update during the training process. Next, CD-MVGNN adopts an aggregation function to produce the graph embedding vector from the node/edge encoder. The process for each module can be categorized into three phases: neighbor aggregation, attached features collection, and message update. As shown in Figure 2.3(a), for the Node-GNN module, taking node v_3 as an example: 1) neighbor nodes aggregation: aggregating the node features of its neighbor nodes v_2, v_4 and v_5; 2) getting the initial edge features as the attached features from the connected edge $\boldsymbol{e}_{23}$, $\mathbf{e}_{34}$, and $\mathbf{e}_{35}$; 3) updating the state of v_3 using (2.3). Figure 2.3(b) demonstrates the edge message construction in the Edge-GNN module. Take edge $\mathbf{e}_{35}$ as an example. 1) neighbor edges aggregation: aggregating the edge features of its neighbor edges, edge $\mathbf{e}_{23}$, edge $\mathbf{e}_{43}$; 2) getting the initial node information as the attached features from the endpoint node v_2 of edge $\mathbf{e}_{23}$, node v_4 of $\mathbf{e}_{43}$; 3) updating the message of $\mathbf{e}_{35}$ using (2.5).



(a) Node-GNN message passing



(b) Edge-GNN message passing

Figure 2.3: Examples of the message passing phase in Node-GNN (2.3(a)) and Edge-GNN (2.3(b)).

Other than the mean-pooling mechanism, we propose to use the *self-attentive aggregation* to learn different weights of the node/edge embeddings to produce the final graph embedding. Furthermore, the self-attentive aggregation layer is shared between the node-central and edge-central encoders, to reinforce the learning of the node features and the edge features, respectively. After the self-attentive aggregation, CD-MVGNN feeds the corresponding graph embeddings to two MLPs to fit the loss function. To stabilize the training process of this multi-view architecture, we employ the *disagreement loss* to enforce the outputs of the two MLPs to be close with each other.

2.4.2 The Node/Edge Feature Extraction of the Molecules

The node/edge feature extraction contains two parts: 1) **node/edge messages**, which are constructed by aggregating neighboring nodes/edges features iteratively; 2) **molecule-level features**, which are the additional molecule-level features generated by RDKit to capture the global molecular information. It consists of 200 features for each molecule [80]. Since we focus on the model architecture part, we follow the exact same protocol of [1] for the initial node (atom) and edge (bond) features selection, as well as the 200 RDKit features generation procedure. The atom features description and size are listed in 5.1, and the bond features are documented in 5.2. The RDKit features are concatenated with the node/edge embedding, to go through the final MLP to make the predictions.

2.4.3 Node-central and Edge-central Encoders

To ease the exposition, in the sequel when using one singe superscript we mean the hop index k while ignoring the layer/iteration index l.

Table 2.1: Atom features [1].

features	size	description
atom type	100	type of atom (e.g., C, N, O), by atomic number
formal charge	5	integer electronic charge assigned to atom
number of bonds	6	number of bonds the atom is involved in
chirality	4	Unspecified, tetrahedral CW/CCW, or other.
number of H	5	number of bonded hydrogen atoms
atomic mass	1	mass of the atom, divided by 100
aromaticity	1	whether this atom is part of an aromatic system
hybridization	5	sp, sp2, sp3, sp3d, or sp3d2

Table 2.2: Bond features [1].

features	size	description
bond type	4	single, double, triple, or aromatic
stereo	6	none, any, E/Z or cis/trans
in ring	1	whether the bond is part of a ring
conjugated	1	whether the bond is conjugated

Node-central Encoder. Node-GNN is built upon the generalized message passing in Equation (2.1). Additionally, we add *input* and *output* layers, to enhance its expressive power. Specifically,

$$\begin{aligned}\mathbf{m}_v^{(k)} &= \mathrm{AGG}_{\mathrm{node}}(\{\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{e}_{uv} \mid u \in \mathcal{N}_v\}), \\ \mathbf{h}_v^{(k)} &= \mathrm{MLP}_{\mathrm{node}}(\{\mathbf{m}_v^{(k)}, \mathbf{h}_v^{(0)}\}),\end{aligned} \tag{2.3}$$

where $\mathbf{h}_v^{(0)} = \sigma(\mathbf{W}_{\mathrm{nin}}\mathbf{x}_v)$ is the input state of Node-GNN, $\mathbf{W}_{\mathrm{nin}} \in \mathbb{R}^{d_{\mathrm{hid}} \times d_n}$ is the input weight matrix. The input layer can also be viewed as a residual connection. After L iterations of message passing, we utilize an additional message passing step with a new weight matrix $\mathbf{W}_{\mathrm{nout}} \in \mathbb{R}^{d_{\mathrm{out}} \times (d_n + d_{\mathrm{hid}})}$ to produce the final node embeddings:

$$\begin{aligned}\mathbf{m}_v^{\mathrm{o}} &= \mathrm{AGG}_{\mathrm{node}}(\{\mathbf{h}_v^{(L,K_L)}, \mathbf{h}_u^{(L,K_L)}, \mathbf{x}_u | u \in \mathcal{N}_v\}), \\ \mathbf{h}_v^{\mathrm{o}} &= \sigma(\mathbf{W}_{\mathrm{nout}}\mathbf{m}_v^{\mathrm{o}}).\end{aligned} \tag{2.4}$$

We denote $\mathbf{H}_n = [\mathbf{h}_1^{\text{o}}, \cdots, \mathbf{h}_p^{\text{o}}] \in \mathbb{R}^{d_{\text{out}} \times p}$ as the output embeddings of Node-GNN, where d_{out} is the dimension of the output embeddings. Figure 2.3(a) in **??** illustrates the process of message passing process in Node-GNN.

Edge-central Encoder. In classical graph theory, the line graph $L(G)$ of a graph G is the graph that encodes the adjacencies between edges of G [81]. $L(G)$ provides a fresh perspective to understand the original graph, i.e., the nodes are viewed as the connections while edges are viewed as entities. Therefore, it enables to perform message passing operation through edges to imitate Node-GNN on $L(G)$ [1]. Namely, given an edge (v, w), we can formulate the Edge-based GNN (Edge-GNN) as:

$$\begin{aligned} \mathbf{m}_{vw}^{(k)} &= \text{AGG}_{\text{edge}}(\{\mathbf{h}_{vw}^{(k-1)}, \mathbf{h}_{uv}^{(k-1)}, \mathbf{x}_u | u \in \mathcal{N}_v \setminus w\}), \\ \mathbf{h}_{vw}^{(k)} &= \text{MLP}_{\text{edge}}(\{\mathbf{m}_{vw}^{(k-1)}, \mathbf{h}_{vw}^{(0)}\}), \end{aligned} \tag{2.5}$$

where $\mathbf{h}_{vw}^{(0)} = \sigma(\mathbf{W}_{\text{ein}}\mathbf{e}_{vw})$ is the input state of Edge-GNN, $\mathbf{W}_{\text{ein}} \in \mathbb{R}^{d_{\text{hid}} \times d_e}$ is the input weight matrix. In Equation (2.5), the state vector is defined on edge $\mathbf{e}_{vw}$ and the neighboring edge set of $\mathbf{e}_{vw}$ is defined as all edges connected to the start node v except the node w. Figure 2.3(b) shows an example of the message passing process in Edge-GNN.

After recurring L steps of message passing, the output of Edge-GNN is the state vectors for edges. In order to incorporate the shared-attentive readout to generate the graph embedding, one more round of message passing on nodes is employed to transform edge-wise embeddings to node-wise embeddings, and generate the second set of node embeddings. Specifically, the Edge-Node transform is established by following,

$$\begin{aligned} \mathbf{m}_v^{\text{o}} &= \text{AGG}_{\text{edge}}(\{\mathbf{h}_{vw}^{(L,K_L)}, \mathbf{h}_{uv}^{(L,K_L)}, \mathbf{x}_u | u \in \mathcal{N}_v\}), \\ \mathbf{h}_v^{\text{o}} &= \sigma(\mathbf{W}_{\text{eout}}\mathbf{m}_v^{\text{o}}), \end{aligned} \tag{2.6}$$

where $\mathbf{W}_{\text{eout}} \in \mathbb{R}^{d_{\text{out}} \times (d_n + d_{\text{hid}})}$ specifies the weight matrix. Therefore, the final output of Edge-GNN provides a new set of *node* embeddings from the edge message passing process. This set of node embeddings are denoted as $\mathbf{H}_e = [\mathbf{h}_1^{\text{o}}, \cdots, \mathbf{h}_p^{\text{o}}] \in \mathbb{R}^{d_{\text{out}} \times p}$.

2.4.4 Cross-Dependent Message Passing Scheme

After constructing the MVGNN framework, we notice that the information flow is not sufficiently efficient, even though the MVGNN model has been proved to have superior performance for many molecular property prediction tasks (as verified in the experiments). Suppose all the information needed to predict the property resides in the molecule itself. For MVGNN, the information flows through two distinct paths in parallel: one path is the node-central encoder, the other one is the edge-central encoder. The information from the two paths finally joins at the disagreement loss.

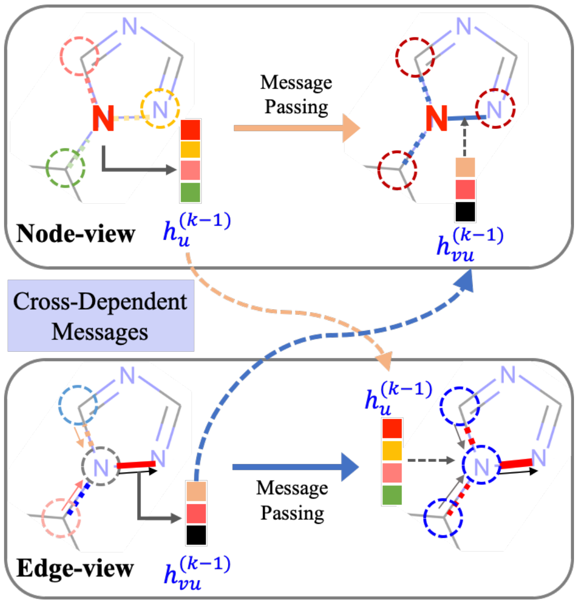


Figure 2.4: Illustration of the cross-dependent message passing scheme. Node message passing utilizes the newest hidden states of neighboring edges, while edge message passing uses the newest hidden states of neighboring nodes. This scheme enables more efficient communication between the two views.

However, the two flows of information could meet *earlier*, to enable more efficient information communication. In pursuit of this point, we propose the *cross-dependent message passing* scheme. On a high level, it makes the message passing operations of the node and edge cross-dependent with each other. Specifically, we change the message passing operations of the node and edge encoders (in (2.3) and (2.5), respectively) to be:

$$\begin{aligned}
\mathbf{m}_v^{(k)} &= \mathrm{AGG}_{\mathrm{node}}(\{\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{h}_{vu}^{(k-1)}, \mathbf{e}_{vu} | u \in \mathcal{N}_v\}),\\
\mathbf{h}_v^{(k)} &= \mathrm{MLP}_{\mathrm{node}}(\{\mathbf{m}_v^{(k)}, \mathbf{h}_v^{(0)}\}),\\
\mathbf{m}_{vw}^{(k)} &= \mathrm{AGG}_{\mathrm{edge}}(\{\mathbf{h}_{vw}^{(k-1)}, \mathbf{h}_{uv}^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{x}_u | u \in \mathcal{N}_v \setminus w\}),\\
\mathbf{h}_{vw}^{(k)} &= \mathrm{MLP}_{\mathrm{edge}}(\{\mathbf{m}_{vw}^{(k)}, \mathbf{h}_{vw}^{(0)}\}).
\end{aligned} \tag{2.7}$$

The first row indicates new node message passing, while the second row shows edge message passing. One can see that when applying aggregation in node message passing, we use the *newest* hidden states of edges (blue colored). While conducting aggregation in edge message passing, it requires the newest hidden states of nodes. In this manner, the two paths of information flow become cross-dependent with each other. Figure 2.4 gives an illustration of this scheme, take the node-view therein for example, the attached edge features $\mathbf{h}_{vu}^{(k-1)}$ used in the current aggregation phase are obtained from the previous step of message passing in the Edge-GNN. Similar approach is performed for the edge-central encoder. Such circulation between the two encoders during every message passing step ensures the information flow stays updated, which empowers CD-MVGNN to be more efficient. We will empirically show that the cross-dependent message passing scheme enables more expressive power compared to the basic MVGNN architecture.

2.4.5 Interpretable Readout

To obtain a fixed length graph representation, a readout component is usually employed on the node embeddings. In this work, we considered two readout transformations to obtain the molecular representation. The first is the simple *mean-pooling readout*, the molecular representation is given by $\xi_n = \frac{1}{p}\sum_{\mathbf{h}_i^o \in \mathbf{H}_n} \mathbf{h}_i^o$.

However, the average operation tends to produce smooth outputs. Therefore, it diminishes the expressive power. To overcome the drawbacks of mean-pooling, we develop the *interpretable shared self-attentive readout component* based on the attention mechanism [46, 82]. Specifically, given an output of node-central encoder $\mathbf{H}_n \in \mathbb{R}^{d_{\text{out}} \times p}$, the self-attention $\mathbf{S}$ over nodes is:

$$\mathbf{S} = \operatorname{softmax}\left(\mathbf{W}_2 \tanh\left(\mathbf{W}_1 \mathbf{H}_n\right)\right), \tag{2.8}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{attn}} \times d_{\text{out}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{r \times d_{\text{attn}}}$ are learnable matrices. In Equation (5.2), $\mathbf{W}_1$ linearly transforms the node embeddings from d_{out}-dimensional space to a h_{attn}-dimensional space. $\mathbf{W}_2$ provides r different insights of node importance, then followed by a softmax function to normalize the importance. To enable the feature information extracted from node and edge encoders communicating during the multi-view training process, we *share* the parameters $\mathbf{W}_1$ and $\mathbf{W}_2$ between the two sub-models. Given $\mathbf{S}$, we can obtain the graph-level embedding by $\boldsymbol{\xi}_n = \mathsf{Flatten}(\mathbf{S}\mathbf{H}_n^\top)$. The self-attention $\mathbf{S}$ implies the importance of the nodes when generating graph embedding, hence indicating contributions of the nodes for downstream tasks, which equips CD-MVGNN with interpretability power.

2.4.6 Disagreement Loss

Suppose the dataset contains graphs $\mathcal{G} = \{G_i\}_{i=1}^K$ and corresponding labels $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^K$. Given one graph G_i, due to the nature of the multi-view architecture, we obtain two graph embeddings $\boldsymbol{\xi}_n$ and $\boldsymbol{\xi}_e$, from the node message passing and edge message passing, respectively. Feeding them into the MLPs results in two predictions $\boldsymbol{\gamma}_{n,i}$ and $\boldsymbol{\gamma}_{e,i}$ for the same target $\mathbf{y}_i$.

Naturally, the losses should get the supervised prediction loss involved, i.e., $\mathcal{L}_{\text{pred}} = \sum_{G_i \in \mathcal{G}} (\mathcal{L}_{\textsf{Node-GNN}}(\mathbf{y}_i, \boldsymbol{\gamma}_{n,i}) + \mathcal{L}_{\textsf{Edge-GNN}}(\mathbf{y}_i, \boldsymbol{\gamma}_{e,i}))$. The specific loss function $\mathcal{L}_{\textsf{Node-GNN}}$ and $\mathcal{L}_{\textsf{Edge-GNN}}$ should depend on the task types, say, cross-entropy for classification and mean squared error for regression.

However, with only the $\mathcal{L}_{\text{pred}}$ loss, we observed unstable behaviors of the training process, which is caused by the loose constraint of the node and edge message passings. To resolve this problem, we propose the *disagreement loss*, which is responsible for restraining the two predictions from node-central and edge-central encoders. Specifically, we employ the mean squared error $\mathcal{L}_{\text{dis}} = \sum_{G_i \in \mathcal{G}} |\gamma_{n,i} - \gamma_{e,i}|^2$. Overall, the shared self-attentive readout and the disagreement loss alleviate the node variant dependency, and reinforce the restriction during the training process to promise the model converge to a stationary status. Finally, the overall loss function contains two parts: $\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda \mathcal{L}_{\text{dis}}$, where λ is a tradeoff hyper-parameter.

2.4.7 Expressive Power of CD-MVGNN

In this section, we theoretically justify the expressive power of CD-MVGNN under the framework of [56], which relates the expressive power of a model to its ability of distinguishing non-isomorphic graphs. By comparing the expressive power with the well justified architecture GIN [56], we reach the following conclusions.

Proposition 1. In terms of distinguishing non-isomorphic graphs under the framework of [56], the following conclusions hold:

1. The basic MVGNN model is at least as powerful as the Graph Isomorphism Network (GIN of [56]), which provably generalizes the WL graph isomorphism test.
2. The CD-MVGNN model is *strictly* more powerful than the Graph Isomorphism Network.

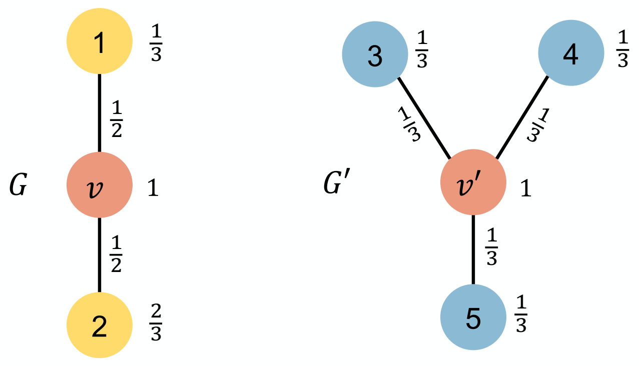


Figure 2.5: Example of the two subgraph structures.

Proof of Proposition 1. Firstly we show that MVGNN is at least as powerful as the Graph Isomorphism Network (GIN of [56]). MVGNN involves both node message passing and edge message passing processes, which constitutes the two-view information flows. Suppose that one blocks the information flowing in the edge passing, say, by setting the initial hidden states of all edges to be 0. At this moment, if one takes the the sum aggregation in [56] as the specific realization of the aggregation operation, then MVGNN recovers the GIN architecture. So we can conclude that MVGNN has at least the same expressive power as GIN.

Then we prove that CD-MVGNN is strictly more powerful than GIN. In order to illustrate this, we construct specific graph examples such that *one* iteration of message passing in GIN cannot distinguish the nodes with different subgraph structures.

However, CD-MVGNN with the cross-dependent message passing scheme is able to discriminate the two nodes. To enable fair comparison, we assume that both GIN and CD-MVGNN use the same aggregation function as in the GIN paper [56]. That is, we use the sum aggregation with a parameter $\epsilon^{(l)}$, i.e., $\mathrm{AGG}^{(l)}(\{\mathbf{h}_v^{(l,k-1)}, \mathbf{h}_u^{(l,k-1)}, \mathbf{e}_{uv} | u \in \mathcal{N}_v\}) = ((1+\epsilon^{(l)})\mathbf{h}_v^{(l,k-1)} + \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(l,k-1)}) || (\sum_{u \in \mathcal{N}_v} \mathbf{e}_{uv})$ ($||$ is the concatenation operation).

Assume there are two graphs G, G' as shown in 2.5. The two nodes therein, v and v' have different local subgraph structures. For simplicity, let all the initial node features, edge features and hidden states have dimensionality as 1. Specifically, for node features and hidden states, we have $\mathbf{h}_v = \mathbf{x}_v = 1$, $\mathbf{h}_{v'} = \mathbf{x}_{v'} = 1$, $\mathbf{h}_1 = \mathbf{x}_1 = \frac{1}{3}$, $\mathbf{h}_2 = \mathbf{x}_2 = \frac{2}{3}$, $\mathbf{h}_3 = \mathbf{h}_4 = \mathbf{h}_5 = \mathbf{x}_3 = \mathbf{x}_4 = \mathbf{x}_5 = \frac{1}{3}$. For edge features and hidden states, one has $\mathbf{h}_{v1} = \mathbf{h}_{1v} = \mathbf{e}_{v1} = \frac{1}{2}$, $\mathbf{h}_{v2} = \mathbf{h}_{2v} = \mathbf{e}_{v2} = \frac{1}{2}$, $\mathbf{h}_{v'3} = \mathbf{h}_{3v'} = \mathbf{e}_{v'3} = \frac{1}{3}$, $\mathbf{h}_{v'4} = \mathbf{h}_{4v'} = \mathbf{e}_{v'4} = \frac{1}{3}$, $\mathbf{h}_{v'5} = \mathbf{h}_{5v'} = \mathbf{e}_{v'5} = \frac{1}{3}$.

Under this setup, we can run one iteration of message passing by hand. Specifically,

— For GIN, suppose its generalized version considers also the initial features. Then the message of node v is $\mathbf{m}_v = (2+\epsilon, 1)$, where the first dimension indicates aggregated node hidden states, the second dimension indicates aggregated edge initial features. The message of node v' is $\mathbf{m}_{v'} = (2+\epsilon, 1)$ as well. Since the state update function is *injective*, after the state update, the hidden states of node v and v' will become the same, thus indistinguishable.

— For CD-MVGNN, one complete iteration of message passing contains one edge message passing and one node message passing. Without loss of generality, let us take it as edge message passing followed by a node message passing.

Consider edge message passing firstly. The edge messages for graph G are:

$$\mathbf{m}_{v2} = (\overbrace{1/3}^{\text{node states sum}}, \overbrace{1+\epsilon/2}^{\text{edge states sum}}, \overbrace{1/3}^{\text{node features sum}}),$$
$$\mathbf{m}_{v1} = (\overbrace{2/3}^{\text{node states sum}}, \overbrace{1+\epsilon/2}^{\text{edge states sum}}, \overbrace{2/3}^{\text{node features sum}}).$$

The two messages are different, the injective state update function will map them into different new states, suppose w.l.o.g. the new states are $\mathbf{h}_{v2} = 1, \mathbf{h}_{v2} = 2$.

The edge messages for graph G' are: $\mathbf{m}_{v'3} = \mathbf{m}_{v'4} = \mathbf{m}_{v'5} =$ $(\overbrace{2/3}^{\text{node states sum}}, \overbrace{1+\epsilon/3}^{\text{edge states sum}}, \overbrace{2/3}^{\text{node features sum}})$. They will be mapped to the same new states, assume they are $\mathbf{h}_{v'3} = \mathbf{h}_{v'4} = \mathbf{h}_{v'5} = 2/3$.

Then consider node message passing with the newest edge hidden states.

$$\mathbf{m}_{v} = (\overbrace{2+\epsilon}^{\text{node states sum}}, \overbrace{3}^{\text{edge states sum}}, \overbrace{1}^{\text{edge features sum}}),$$
$$\mathbf{m}_{v'} = (\overbrace{2+\epsilon}^{\text{node states sum}}, \overbrace{2}^{\text{edge states sum}}, \overbrace{1}^{\text{edge features sum}}).$$

Now we have different messages for nodes v and v', so it will be mapped to different new hidden states by the injective multi-layer perceptron (MLP). Thus the two nodes become distinguishable under the cross-dependent message passing scheme of CD-MVGNN. $\square$

Proposition 1 shows that CD-MVGNN model has sufficient model capacity in terms of distinguishing graphs compared to the GIN architecture and the WL graph isomorphism test. This observation also explains why it reaches superior performance on various baseline tasks, which will be further verified in the experiments.

2.5 Experiments

We conduct performance evaluations of MVGNN and CD-MVGNN with various SOTA baselines on molecular property classification and regression tasks. We also preform ablation studies on different components of our models. Lastly, we conduct case studies to demonstrate the interpretability power of the proposed models. Source code will be released soon.

2.5.1 Experimental Setup

2.5.1.1 Description of Dataset

We experimented with 11 popular benchmark datasets, among which six are classification tasks and the others are regression tasks. Table 5.3 summaries the dataset statistics [66], including the property category, number of tasks and evaluation metrics of all datasets. Six datasets are used for classification, and five datasets for regression. Noted, ToxCast contains 617 tasks, which makes it extremely time consuming to apply N-Gram model, since N-Gram requires task-based preprocess.

Table 2.3: Datasets statstics.

Category	Dataset	Task	# Tasks	# Molecules	Metric
Biophysics	BACE	Classification	1	1513	AUC-ROC
Physiology	BBBP	Classification	1	2039	AUC-ROC
	Tox21	Classification	12	7831	AUC-ROC
	ToxCast	Classification	617	8576	AUC-ROC
	SIDER	Classification	27	1427	AUC-ROC
	ClinTox	Classification	2	1478	AUC-ROC
Quantum Mechanics	QM7	Regression	1	6830	MAE
	QM8	Regression	12	21786	MAE
Physical Chemistry	ESOL	Regression	1	1128	RMSE
	Lipophilicity	Regression	1	4200	RMSE
	FreeSolv	Regression	1	642	RMSE

Molecular Classification Datasets. BACE dataset is collected for recording compounds which could act as the inhibitors of human β-secretase 1 (BACE-1) in the past few years [83]. The Blood-brain barrier penetration (BBBP) dataset contains the records of whether a compound carries the permeability property of penetrating the blood-brain barrier [78]. Tox21 and ToxCast [77] datasets include multiple toxicity labels over thousands of compounds by running high-throughput screening test on thousands of chemicals . SIDER documents marketed drug along with its adverse drug reactions, also known as the Side Effect Resource [84]. ClinTox dataset compares drugs approved through FDA and drugs eliminated due to the toxicity during clinical trials [85].

Molecular Regression Datasets. QM7 dataset is a subset of GDB-13, which records the computed atomization energies of stable and synthetically accessible organic molecules, such as HOMO/LUMO, atomization energy, etc. It contains various molecular structures such as triple bonds, cycles, amide, epoxy, etc [86]. QM8 dataset contains computer-generated quantum mechanical properties, e.g., electronic spectra and excited state energy of small molecules [79]. Both QM7 and QM8 contain 3D coordinates of the molecules along with the molecular SMILES. ESOL documents the solubility of compounds [87]. Lipophilicity dataset is selected from ChEMBL database, which is an important property that affects the molecular membrane permeability and solubility. The data is obtained via octanol/water distribution coefficient experiments [88]. FreeSolv dataset is selected from the Free Solvation Database, which contains the hydration free energy of small molecules in water from both experiments and alchemical free energy calculations [89].

2.5.1.2 Dataset Splitting and Experimental Setting.

We apply the *scaffold splitting* for all tasks on all datasets, which is more practical and challenging than random splitting. Random splitting is a common process to split the dataset into train, validation and test set randomly. However, it does not simulate the real-world scenarios for evaluating molecule-related machine learning methods [90]. Scaffold splitting splits the molecules with distinct two-dimensional structural frameworks into different subsets [90], e.g. molecules with benzene ring would be split into one subset, which could be the train/validation/test set. This means that the validation/test dataset might contain molecules with unseen structures from the training dataset, which makes the learning much more difficult. Yet, it is more difficult for the learning algorithm to accomplish satisfactory performance, but from the chemistry perspective, it is more meaningful and consequential for molecular property prediction. To alleviate the effects of randomness and over-fitting, as well as to boost the robustness of the experiments, we apply cross-validation on all the experiments. All of our experiments run 10 randomly-seeded 8:1:1 data splits, which follows the same protocols of [1].

2.5.1.3 Baselines

We thoroughly evaluate the performance of our methods against popular baselines from both machine learning and chemistry communities. Among them,

- Influence Relevance Voting (IRV) is a K-Nearest Neighbor classifier, which assumes similar sub-structures reveal similar functionality [64].
- LogReg [91] predicts the binary label by learning the coefficient combination of a logistic function based on the input features.
- GraphConv [34] is the vanilla graph convolutional model implementation by updating the atom features with its neighbor atoms' features. Compared with GraphConv, Weave [67] model updates the atom features by constructing atom-pair with all other atoms, then combining the atom-pair features.
- SchNet [69] and MGCN [76] explore the molecular structure by utilizing the physical information, the 3D coordinates of each atom.

- N-Gram [92] proposes an unsupervised method to enhance the molecular representation learning by exploiting special attribute structure.
- AttentiveFP [71] exploits graph attention mechanism to learn molecular representations.
- CMPNN utilizes the information from both atoms and the corresponding incoming bonds.
- GIN [56] is the implementation of Graph Isomorphism Network.
- MPNN [73] and DMPNN [1] perform the message passing scheme on atoms and bonds, respectively.

For SchNet, MGCN and AttentiveFP, we use DGL [93] implementations; for N-Gram, CMPNN, GIN and DMPNN, we use open source codes provided by the authors; for MPNN, we use the implementation by [1]; for others, we use the MoleculeNet [66] implementations. IRV and LogReg are available for classification tasks only on MoleculeNet [66].

2.5.1.4 Evaluation Metrics

All classification tasks are evaluated by AUC-ROC. For the regression tasks, we apply MAE and RMSE to evaluate the performance on different datasets.

2.5.1.5 Two Naive Schemes Setup

To demonstrate the effectiveness of the shared self-attentive readout and the disagreement loss in the multi-view architecture, we also implement two naive schemes. Concat + Mean concatenates the mean-pooling outputs of the two sub-modules, and Concat + Attn concatenates the self-attentive outputs [3]of the two sub-modules.

[3]We do not share the attention here.

2.5.1.6 Hyper-parameters

We adopt Adam optimizer for model training. We use the Noam learning rate scheduler with two linear increase warm-up epochs and exponential decay afterwards [94].

For proposed models on each dataset, we try different hyper-parameter combinations via random search, and take the hyper-parameter set with the best test score. To remove randomness, we conduct experiments 10 times with different seeds, along with the best-parameter set, to get the final result. The details of the hyper-parameters of the implementation of our models are introduced in Table 2.4.

Table 2.4: Hyper-parameter Description.

Hyper-parameter	Description	Range
init_lr	initial learning rate of Adam optimizer and Noam learning rate scheduler	0.0001˜0.0004
max_lr	maximum learning rate of Noam learning rate scheduler	0.001˜0.004
final_lr	final learning rate of Noam learning rate scheduler	0.0001˜0.0004
depth	number of the message passing hops (K)	2˜6
hidden_size	number of the hidden dimensionality of the message passing network in two encoders (d_{hid})	7˜19
dropout	dropout rate	0.5
weight_decay	weight decay percentage for Adam optimizer	0.00000001˜0.000001
ffn_num_layers	number of the MLPs	2˜4
ffn_hidden_size	number of the hidden dimensionality in the MLP	7˜19
bond_drop_rate [95]	random remove certain percent of edges	0˜0.6
attn_hidden	number of hidden dimensionality in the self-attentive readout (d_{attn})	32˜256
attn_out	number of output dimensionality in the self-attentive readout (r)	1˜8
dist_coff	the coefficient of the disagreement loss (λ)	0.01˜0.2

2.5.2 Performance on Classification Tasks

Table 2.5 summarizes the results of the classification tasks. To evaluate the robustness of our method, we report the mean and standard deviation of 10 times runs with different random seeds for MVGNN, CD-MVGNN and the other models. Table 2.5 indicates the following: (1) our MVGNN and CD-MVGNN models gain significant

[4]Result not presented since N-Gram requires task-based preprocessing, which cannot be finished in reasonable days.

Table 2.5: Performance of classification tasks on AUC-ROC (higher is better) with the scaffold split. Best score is marked as **bold**, and the best baseline is marked in gray. Green cells indicate the results of our methods.

Method	BACE	BBBP	Tox21	ToxCast	SIDER	ClinTox
IRV	$0.838_{\pm 0.055}$	$0.877_{\pm 0.051}$	$0.699_{\pm 0.055}$	$0.604_{\pm 0.037}$	$0.595_{\pm 0.022}$	$0.741_{\pm 0.069}$
LogReg	$0.844_{\pm 0.040}$	$0.835_{\pm 0.067}$	$0.702_{\pm 0.028}$	$0.613_{\pm 0.033}$	$0.583_{\pm 0.034}$	$0.733_{\pm 0.084}$
GraphConv	$0.854_{\pm 0.011}$	$0.877_{\pm 0.036}$	$0.772_{\pm 0.041}$	$0.650_{\pm 0.025}$	$0.593_{\pm 0.035}$	$0.845_{\pm 0.051}$
Weave	$0.791_{\pm 0.008}$	$0.837_{\pm 0.065}$	$0.741_{\pm 0.044}$	$0.678_{\pm 0.024}$	$0.543_{\pm 0.034}$	$0.823_{\pm 0.023}$
SchNet	$0.750_{\pm 0.033}$	$0.847_{\pm 0.024}$	$0.767_{\pm 0.025}$	$0.679_{\pm 0.021}$	$0.545_{\pm 0.038}$	$0.717_{\pm 0.042}$
MGCN	$0.734_{\pm 0.030}$	$0.850_{\pm 0.064}$	$0.707_{\pm 0.016}$	$0.663_{\pm 0.009}$	$0.552_{\pm 0.018}$	$0.634_{\pm 0.042}$
N-Gram	$0.876_{\pm 0.035}$	$0.912_{\pm 0.013}$	$0.769_{\pm 0.027}$	$-$[4]	$0.632_{\pm 0.005}$	$0.855_{\pm 0.037}$
AttentiveFP	$0.863_{\pm 0.015}$	$0.908_{\pm 0.050}$	$0.807_{\pm 0.020}$	$0.579_{\pm 0.001}$	$0.605_{\pm 0.060}$	$0.933_{\pm 0.020}$
CMPNN	$0.869_{\pm 0.023}$	$0.929_{\pm 0.025}$	$0.810_{\pm 0.022}$	$0.709_{\pm 0.006}$	$0.617_{\pm 0.016}$	$0.922_{\pm 0.017}$
GIN	$0.845_{\pm 0.040}$	$0.894_{\pm 0.011}$	$0.811_{\pm 0.028}$	$0.703_{\pm 0.006}$	$0.591_{\pm 0.039}$	$0.869_{\pm 0.076}$
MPNN	$0.815_{\pm 0.044}$	$0.913_{\pm 0.041}$	$0.808_{\pm 0.024}$	$0.691_{\pm 0.013}$	$0.595_{\pm 0.030}$	$0.879_{\pm 0.054}$
DMPNN	$0.852_{\pm 0.053}$	$0.919_{\pm 0.030}$	$0.826_{\pm 0.023}$	$0.718_{\pm 0.011}$	$0.632_{\pm 0.023}$	$0.897_{\pm 0.040}$
Concat + Mean	$0.842_{\pm 0.004}$	$0.930_{\pm 0.002}$	$0.816_{\pm 0.003}$	$0.721_{\pm 0.001}$	$0.621_{\pm 0.007}$	$0.882_{\pm 0.008}$
Concat + Attn	$0.832_{\pm 0.007}$	$0.931_{\pm 0.006}$	$0.819_{\pm 0.003}$	$0.728_{\pm 0.002}$	$0.632_{\pm 0.008}$	$0.913_{\pm 0.009}$
MVGNN	$0.863_{\pm 0.002}$	$\mathbf{0.938}_{\pm 0.003}$	$0.833_{\pm 0.001}$	$0.729_{\pm 0.006}$	$\mathbf{0.644}_{\pm 0.003}$	$0.930_{\pm 0.003}$
CD-MVGNN	$\mathbf{0.892}_{\pm 0.011}$	$0.933_{\pm 0.006}$	$\mathbf{0.836}_{\pm 0.006}$	$\mathbf{0.744}_{\pm 0.005}$	$0.639_{\pm 0.012}$	$\mathbf{0.945}_{\pm 0.017}$

enhancement against SOTAs on all datasets consistently, CD-MVGNN performs slightly better than MVGNN with a 1.80% average AUC boost compared with the SOTAs on each dataset, which is regarded as the remarkable boost, considering the challenges on these classification benchmarks with scaffold splitting method. (2) Compared with the SOTAs, MVGNN and CD-MVGNN has much smaller standard deviation, which implies that our models are more robust than the baselines. (3) Compared with the two simple variants, MVGNN and CD-MVGNN demonstrate the superiority both on performance and robustness. It validates the effectiveness of the multi-view architecture with self-attentive readout and disagreement loss constraints. Moreover, Figure 2.7(a) demonstrates the improvement between our models and the best SOTA model on the classification tasks, according to Table2.5. As observed, proposed models are able to achieve up to 3.62% on the ToxCast dataset.

Table 2.6: The performance comparison of regression task on scaffold split (smaller is better). The evaluation metric of QM7/QM8 is MAE and that of ESOL/Lipo/FreeSolv is RMSE.

Method	QM7	QM8	ESOL	Lipo	FreeSolv
GraphConv	118.875 $\pm$0.219	0.021 $\pm$0.001	1.068 $\pm$0.050	0.712 $\pm$0.049	2.900 $\pm$0.135
Weave	94.688 $\pm$2.705	0.022 $\pm$0.001	1.158 $\pm$0.055	0.813 $\pm$0.042	2.398 $\pm$0.250
SchNet	74.204 $\pm$4.983	0.020 $\pm$0.002	1.045 $\pm$0.064	0.909 $\pm$0.098	3.215 $\pm$0.755
MGCN	77.623 $\pm$4.734	0.022 $\pm$0.002	1.266 $\pm$0.147	1.113 $\pm$0.041	3.349 $\pm$0.097
N-Gram	125.630 $\pm$1.480	0.032 $\pm$0.003	1.100 $\pm$0.160	0.876 $\pm$0.033	2.512 $\pm$0.190
AttentiveFP	126.690 $\pm$4.020	0.028 $\pm$0.001	0.853 $\pm$0.060	0.650 $\pm$0.030	2.030 $\pm$0.420
CMPNN	75.875 $\pm$12.360	0.015 $\pm$0.002	0.838 $\pm$0.14	0.625 $\pm$0.027	2.060 $\pm$0.505
GIN	74.388 $\pm$4.543	0.430 $\pm$0.001	1.009 $\pm$0.065	0.690 $\pm$0.074	2.620 $\pm$0.840
MPNN	112.960 $\pm$17.211	0.015 $\pm$0.002	1.167 $\pm$0.430	0.672 $\pm$0.051	2.185 $\pm$0.952
DMPNN	105.775 $\pm$13.202	0.0143 $\pm$0.002	0.980 $\pm$0.258	0.653 $\pm$0.046	2.177 $\pm$0.914
Concat + Mean	72.532 $\pm$2.657	0.0129 $\pm$0.0005	0.806 $\pm$0.040	0.610 $\pm$0.024	2.003 $\pm$0.317
Concat + Attn	73.132 $\pm$3.845	0.0128 $\pm$0.0005	0.809 $\pm$0.043	0.601 $\pm$0.015	2.026 $\pm$0.227
MVGNN	71.325 $\pm$2.843	0.0127 $\pm$0.0005	0.8049 $\pm$0.036	0.599 $\pm$0.016	1.840 $\pm$0.194
CD-MVGNN	**70.358** $\pm$5.962	**0.0124** $\pm$0.001	**0.779** $\pm$0.026	**0.553** $\pm$0.013	**1.552** $\pm$0.123

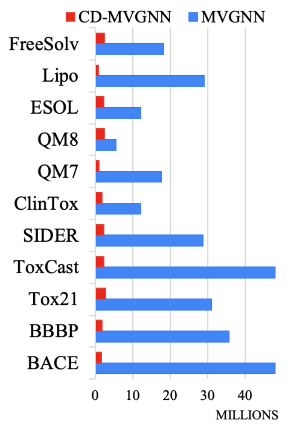
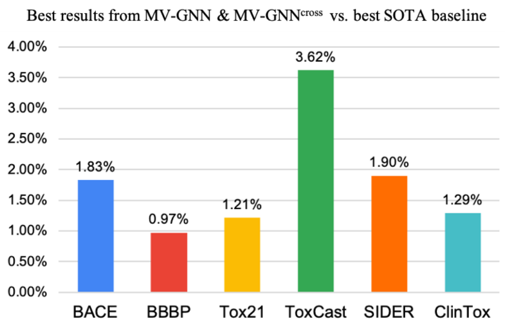


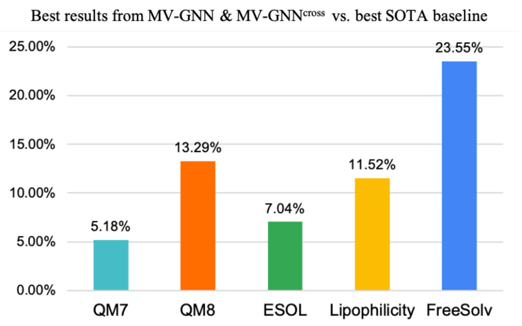
Figure 2.6: Model parameters comparison.

2.5.3 Performance on Regression Tasks

Table 2.6 reports the results of MVGNN and CD-MVGNN on regression tasks over 5 benchmark datasets and 9 baseline models. As we can see, our models achieve the best performance on the regression tasks too. 2.7(b) illustrates the relative improvement from our model with other SOTAs. Recent graph studies on molecules generally focus on certain areas which lack universality. For example, SchNet and MGCN perform good on quantum mechanics datasets (QM7 and QM8) since they utilizes the distances between atoms using the 3D coordinate information, but cannot capture sufficient molecule-level information to generate informative molecular representations. On the other hand, proposed models consistently achieve remarkable performance over all datasets. Specifically, CD-MVGNN relatively improves **23.55%** over other models on the FreeSolv dataset, yet again, reveals the superiority and robustness of the multi-view architecture. The results of Concat + Mean and Concat + Attn on regression datasets also prove the effectiveness of MVGNN.



(a) Classification tasks.



(b) Regression tasks.

Figure 2.7: Improvement visualization between best proposed models with the best SOTA model on classification tasks (2.7(a)) and regression tasks (2.7(b)).

2.5.4 Ablation Studies on Key Design Choices

This section focuses on the impacts of key components in our proposed models.

2.5.4.1 Cross-dependent message passing.

Table 2.7 compares the parameter size of MVGNN and CD-MVGNN for each dataset. As observed, CD-MVGNN demands significantly less parameters. We plot the average number of parameters in the MVGNN and CD-MVGNN models in Figure 5.7, these are the models with the best performance in each hyerparameter search. It clearly indicates that CD-MVGNN, while enjoying competitive performance, needs *much less* amount of parameters than MVGNN. Specifically, the average number of parameters of MVGNN is *15.26* times of that of CD-MVGNN. This confirms that the *cross-dependent message passing* scheme can significantly improve the expressive power of the model, by enabling a more efficient information communication scheme in the multi-view architecture.

Table 2.7: Number of model parameters.

Dataset	MVGNN	CD-MVGNN
BACE	47,979,250	1,655,602
BBBP	35,727,858	1,736,002
Tox21	31,057,826	2,757,024
ToxCast	47,877,458	2,358,034
SIDER	28,710,226	2,262,054
ClinTox	12,106,114	1,812,804
QM7	17,628,514	1,029,602
QM8	5,493,314	2,418,424
ESOL	12,106,114	2,248,802
Lipophilicity	29,069,026	826,402
FreeSolv	18,266,626	2,457,602

2.5.4.2 Self-attentive readout and disagreement loss.

We report the results of three datasets with fixed train/valid/test sets to evaluate the impacts in Table 2.8, which demonstrates the proposed multi-view models overall performs the best on all three datasets. Moreover, we find that both attention and disagreement loss can boost the performance compared with "No All" method. Particularly, when the self-attention mechanism is employed, the performance has a significant boost, which proves that the molecular property is affected by the various atoms differently. Hence, the weights of atoms should not be considered equivalently. Thus, the proposed MVGNN and CD-MVGNN models that adopt both disagreement loss and self-attention outperforms the other variants, indicating that the combination of them would significantly facilitate the model training.

2.5.5 Visualization of Interpretability Results

To illustrate the interpretability power of proposed models, we visualize certain molecules with the learned attention weights of CD-MVGNN associated with each atom

	ToxCast	SIDER	ClinTox
No All	0.718	0.644	0.852
Only Attention	0.728	0.646	0.901
Only Disagreement Loss	0.722	0.648	0.863
MVGNN	**0.731**	**0.652**	**0.907**
CD-MVGNN	**0.744**	**0.657**	**0.923**

Table 2.8: Ablation study on the variants of CD-MVGNN.

within one molecule from the Clintox dataset, with toxicity as the labels. Figure 2.8 instantiates the graph structures of the molecules along with the corresponding atom attentions. The attention values lower than 0.01 are omitted. We observe that different atoms indeed react distinctively: 1) Most carbon (C) atoms that are responsible for constructing the molecule topology have got zero attention value. It is because these kinds of sub-structures usually do not affect the toxicity of a compound. 2) Beyond that, CD-MVGNN promotes the learning of the functional groups with impression on molecular toxicity, e.g., toxic functional group *trifluoromethyl* and *cyanide* are known responsible for the toxicity [96], which reveal extremely high attention value in Figure 2.8. These high attention values can be used to explain the toxicity of the molecules. Compared with the previous models, CD-MVGNN is able to provide reasonable interpretability results for the predictions, which is crucial for the real drug discovery.

Furthermore, we provide a comprehensive statistics of the attention values over the entire ClinTox dataset. Figure 2.9 demonstrates the average attention values and the total occurrences of each element. It is notable that, 1) atoms with high frequency do not receive high attention. For example, atom C is an essential element to maintain the molecular topology, yet it does not have significant impact on the toxicity. 2) atoms with low frequency but high attention values are generally heavy elements. For

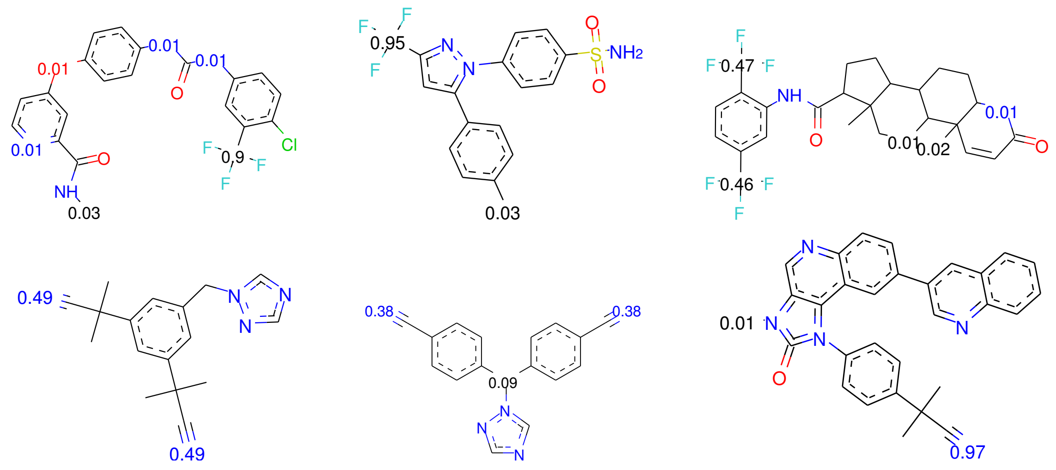


Figure 2.8: Visualization of attention values on ClinTox data. Attention value smaller than 0.01 is omitted. Different color indicates different elements: black: C, blue: N, red: O, green: Cl, yellow: S, sky-blue: F. First row: the molecules with trifluoromethyl. Second row: the molecules with cyanide.

example, Hg (Mercury) is widely known by its toxicity. The accompanied attention value of Hg is relevantly high because it usually affects the toxic property greatly. Overall, the case study shows that the proposed CD-MVGNN model is able to provide reasonable interpretability for the prediction results.

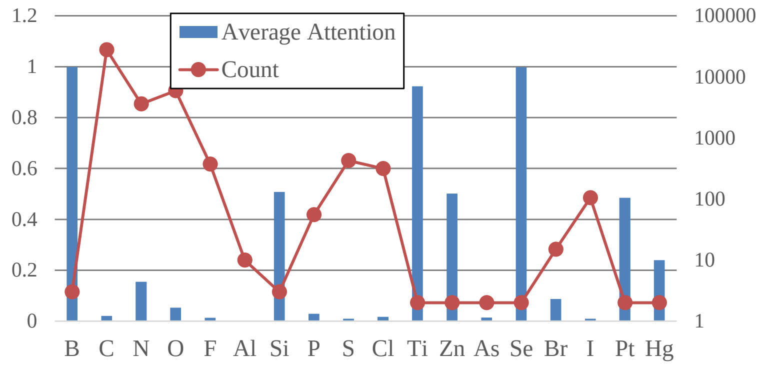


Figure 2.9: Statistics of attentions in ClinTox. Left axis: the average attention value of the element. Right axis: the count of the element.

2.6 Conclusions

We propose a novel cross-dependent graph neural network (CD-MVGNN) for molecular property prediction, which is deployed via a multi-view architecture (MVGNN). Unlike previous attempts focusing exclusively on either atom-oriented graph structures or bond-oriented graph structures, our method, inspired by multi-view learning, takes both atom and bond information into consideration. Most importantly, we develop a cross-dependent message passing scheme to allow concurrent circulation between the two views during the training in CD-MVGNN. Such approach ensures the information flow stay updated for each GNN aggregation step, which boost the efficiency of generalized GNN, as well as increase the expressive power of CD-MVGNN. Extensive experiments against SOTA models demonstrate that proposed models outperform all baselines significantly, as well as equip with strong robustness.

CHAPTER 3

GRADIENT-NORM BASED ATTENTIVE LOSS FOR MOLECULAR PROPERTY PREDICTION

Many studies have addressed molecular property prediction by designing deep learning algorithms, e.g., sequence-based models and graph-based models. However, the underlying data distribution is rarely explored. We discover that there exist easy samples and hard samples in the molecule datasets, and the overall distribution is usually imbalanced. Current research mainly treats them equally during the model training, while we believe that they shall not share the same weights since neural network training is dominated by the majority class. Therefore, we propose to utilize a self-attention mechanism to generate a learnable weight for each data sample according to the associated gradient norm. The learned attention value is then embedded into the prediction models to construct an attentive loss for network updating and back-propagation. It is empirically demonstrated that our proposed method can consistently boost the prediction performance for both classification and regression tasks.

3.1 Introduction

Molecular property prediction is crucial to drug discovery since it helps determine the functions of new drugs. To date, machine learning techniques, especially deep learning methods, have been widely and successfully used in many fields, e.g., computer vision (CV) [8, 9, 11, 13, 97], natural language processing (NLP) [18, 37, 98], and bioinformatics [24, 29, 43, 44]. It is natural to apply deep learning on molecular property prediction too. Many studies have attempted to address molecular property prediction problem by utilizing and designing deep learning algorithms [27, 73, 99]. A molecule can be represented as either a sequence string (SMILES representation), or a graph structure. Therefore, sequence-based models used in NLP can be employed to predict the molecular property [100, 101], while graph-based models can be utilized on the molecular graph structure [1, 35, 102, 103].

The molecules naturally contain some characteristics according to their biological structures, which leads to an inconsistency during the training of deep learning models. In specific, the properties of some molecules are easier to predict since they may have simpler structures or typical functional groups, while some molecules are difficult to predict since they may contain identical sub-structures but express opposite properties. The hard cases usually result in a bad prediction performance. Most of the commonly used datasets for molecular property prediction generally include more easy samples than hard samples, which brings in an imbalance problem for model training. Neural networks cannot harmonize such easy and hard cases perfectly since most of them adopt a batch learning method, which will be dominant by the majority class. In order to pursue better overall performance, neural networks are trained to minimize the comprehensive loss, which may leads to a result that easy samples are learned better but hard samples are barely learned. Besides, the gradient updates for those easy samples are quite little which cannot contribute much to the model training.

Similar problems have been studied in the image detection area since the target objects are usually difficult to detect due to the majority of background contexts. Several researches have addressed such problems by designing algorithms to balance the loss between easy samples and hard samples [104–107]. However, the molecular property prediction problem is more complicated. Unlike image detection where the target objects can be easily observed by human beings, hard samples in molecule data are impractical to recognize. Moreover, it is possible that a molecule is an easy sample for some properties but a hard sample for other properties, such a scenario may varies for different models too. Therefore, we propose an attentive loss function to enable the neural network to learn a weight for each sample according to the prediction difficulty level.

The intuition of our method is that easy samples and hard samples should be treated differently. Rather than simply assigning larger weights for hard samples and smaller weights for easy samples, a feasible algorithm should be designed. The reason is that it is not always good to enforce the network to learn hard samples, sometimes those may be outliers or extremely complicated molecules. Thus, we first calculate the relative gradient norm for each data sample to represent the prediction difficulty level, then employ a self-attention mechanism to allow the network to learn a proper weight for each sample. The learned attention values are then embedded with the prediction loss for model updating and back-propagation. The contribution of the proposed method can be summarized as: 1) to the best of our knowledge, we are the first to propose a learnable weighted loss to tackle the easy-hard sample imbalance problem; 2) extensive experiments on molecular property prediction tasks demonstrate that models with proposed attentive loss promote the prediction performance consistently; 3) our method is not limited to the prediction tasks or the format of input data. It can be easily embedded into any supervised models with any input data, e.g., sequence-based protein structure prediction, image-based medical image classification.

3.2 Related Work

3.2.1 Molecule Encoder Models for Property Prediction

One crucial part of addressing the molecular property prediction problem is to get an accurate vector representation of the molecule. Since the molecules can

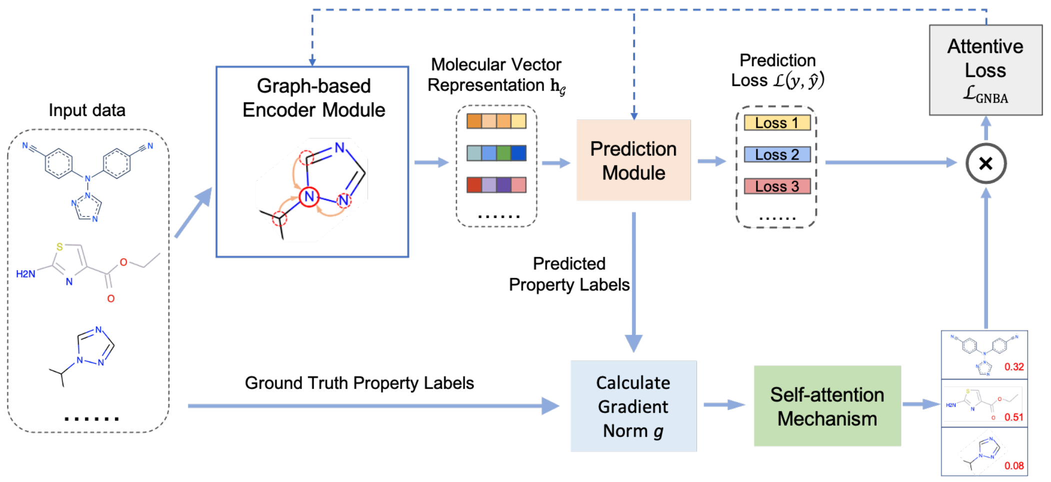

Input data
Graph-based Encoder Module
Molecular Vector Representation $\mathbf{h}_{\mathcal{G}}$
Prediction Module
Prediction Loss $\mathcal{L}(y,\hat{y})$
Loss 1
Loss 2
Loss 3
Attentive Loss $\mathcal{L}_{\mathrm{GNBA}}$
Predicted Property Labels
Ground Truth Property Labels
Calculate Gradient Norm g
Self-attention Mechanism
0.32
0.51
0.08

Figure 3.1: Overview of the model framework. The molecules are fed into the graph-based encoder module to generate vector representations. The vector representations are then used in the prediction module to predict the property labels. Next, the gradient norms are calculated by the predictions and the ground truth property labels, and then go through a self-attention mechanism to generate the attention values for the molecules. Last, the attentions are embedded into the prediction loss for model updating and back-propagation.

be represented as SMILES sequences or graphs, the encoder models can be either sequence-based models or graph-based models. Sequence-based models spot the potentially useful information of the molecular SMILES sequence data by adequately training them using Recurrent Neural Networks (RNNs), in order to obtain the vector representation of the molecule [39,100,101]. Graph-based techniques are used to utilize the graph structure of a molecule, and Graph Neural Networks (GNNs) are employed to generate the molecular representation by embedding the graph features into a continuous vector [1,56,73,102,103,108]. Graph Isomorphism Network (GIN) [56] and Graph Attention Network (GAT) [108] are two representative work. In this paper, we take the graph-based models as the backbone models to verify the effectiveness of our method.

3.2.2 Loss Function Regarding Class Imbalance

Several studies have attempted to harmonize the imbalanced data distribution problem. Focal loss discovers the positive-negative sample imbalance problem in the image detection area [104]. The background negative samples are much more than the positive target samples, thus the regular dense sampling method overwhelms the model training. They propose to reshape the standard cross-entropy loss to assign smaller weights for those well-classified samples. Later, [106] points out the easy-hard sample imbalance problem for detection. They summarize the disharmonies with regards to the distribution of the gradient norm and design a gradient harmonizing mechanism (GHM) to modify the gradients by reformulating the loss function.

3.3 Methodology

The implementation of our proposed method is introduced in this section. The overview of the entire model framework is illustrated in Fig. 3.1.

3.3.1 Molecular Property Prediction Model

3.3.1.1 Problem Definition

The molecular property prediction problem is a prediction task that includes classification and regression problem. Given a molecule $\mathcal{M}$, it contains property y, where $y \in \{0, 1\}$ for classification problem and $y \in \mathbb{R}$ for regression problem. The commonly used representation of molecule $\mathcal{M}$ refers to either a sequence or a graph. For sequence-based input, $\mathcal{M}$ is represented by SMILES, and language models are applied to convert the SMILES string to For graph-based input, the graph structure of $\mathcal{M}$ is usually extracted by RDKit [80]. Then GNN-based models are employed to learn the vector representation $\mathbf{h}_g \in \mathbb{R}^{d_g}$ according to the graph features, where

d_g is the dimension of graph-based features. In our experiments, we take the graph structure of $\mathcal{M}$ as the input and take GIN [56] and GAT [108] as the backbone models to conduct extensive experiments.

3.3.1.2 Graph-based Encoder Module

Molecule $\mathcal{M}$ can be naturally represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $|\mathcal{V}| = p$ refers to the set of p atoms and $|\mathcal{E}| = q$ refers to a set of q bonds within the molecule. The features of atom v is referred as $\mathbf{a}_v \in \mathbb{R}^{d_a}$, and the features of bond (v, u) is referred as $\mathbf{b}_{vu} \in \mathbb{R}^{d_b}$, where $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_b}$ represent the feature dimension of atom and bond respectively. $\mathcal{N}(v)$ denotes the neighbor atoms of atom v, which is identified by the bonds between atoms.

Most of the commonly used GNN-based models follow a procedure of message passing and state update. In specific, the state of the target node v at layer/iteration l $(l = 0, 1, \ldots, L)$ is updated by aggregating the information of its neighborhood $\mathbf{h}_u$ $(u \in \mathcal{N}(v))$, then combined with the state of itself $\mathbf{h}_v$. After L layer/iteration, the states of all the nodes are captured to generate a vector representation $\mathbf{h}_\mathcal{G}$ through a readout mechanism. The process can be formulated as:

$$\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \text{AGGREGATE}_{\ l} \left(\left\{ \mathbf{h}_u^{(l-1)}, \forall u \in \mathcal{N}(v) \right\} \right), \tag{3.1}$$

$$\mathbf{h}_v^{(l)} = \sigma \left(W^{(l)} \cdot \text{CONCAT} \left(\mathbf{h}_v^{(l-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(l)} \right) \right) , \tag{3.2}$$

$$\mathbf{h}_\mathcal{G} = \text{READOUT}(\{\mathbf{h}_v^{(L)} |\ v \in \mathcal{V}\}), \tag{3.3}$$

where $W^{(l)}$ is the weight matrix, and σ is the activation function. The readout mechanism can be operations like summation or mean.

In this paper, we use two commonly used variants of graph-based models, GIN and GAT, as the backbone models to confirm the effectiveness of proposed method.

GIN is theoretically proved as one of the most powerful GNN models. It utilizes multi-layer perceptron (MLP) for state update, as well as employ a concatenate operation over all layers/iterations during the readout phase. The updated rule in Equation 3.2 and 3.3 can be summarized as:

$$\mathbf{h}_v^{(l)} = \mathrm{MLP}^{(l)} \left(\left(1 + \epsilon^{(l)}\right) \cdot \mathbf{h}_v^{(l-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(l-1)} \right), \tag{3.4}$$

$$\mathbf{h}_{\mathcal{G}} = \mathrm{CONCAT}\left(\mathrm{READOUT}\left(\left\{\mathbf{h}_v^{(l)} \mid v \in \mathcal{V}\right\}\right)\right), \tag{3.5}$$

where $l = 0, 1, \ldots, L$, and ϵ is a fixed scalar or a learnable parameter.

GAT employs an attention mechanism over the neighbors of target node, thus each neighbor node gets an associated weight, e.g., more important nodes receive higher weight values. Rather than treating every neighborhood equally, GAT considers the learned attention a_{vu} along with every neighborhood during the aggregation part. Therefore, Equation 3.2 can be updated with

$$\mathbf{h}_v^{(l)} = \sigma \left(\sum_{u \in N(v)} a_{vu} W^{(l-1)} \mathbf{h}_u^{(l-1)} \right), \tag{3.6}$$

$$a_{vu} = \exp \left(\frac{\sigma\left(\boldsymbol{\alpha}^{\mathrm{T}} \left[W\mathbf{h}_v \| W\mathbf{h}_u\right]\right)}{\sum_{k \in N(v)} \boldsymbol{\alpha}^{\mathrm{T}} \left[W\mathbf{h}_v \| W\mathbf{h}_k\right]} \right), \tag{3.7}$$

where $\boldsymbol{\alpha}$ is a weight vector parameter for the attention mechanism, and $(\cdot)^T$ denotes the transposition and $\|$ represents the concatenation operation.

3.3.1.3 Prediction Module

After going through the graph encoder module, a graph representation $\mathbf{h}_{\mathcal{G}}$ is obtained and fed into the following inference module for property prediction. The prediction module can be simply one or several fully connected (FC) layers or MLP. Here we follow the same protocol used in [102], which is one FC layer:

$$\hat{y} = FC(\mathbf{h}_{\mathcal{G}}). \tag{3.8}$$

$\hat{y}$ is the output of the prediction module, which refers to the predicted probability for the classification tasks and the actual predicted property value for the regression tasks. Furthermore, we defined the supervised loss $\mathcal{L}(y, \hat{y})$ as $\mathcal{L}_{cls}$ and $\mathcal{L}_{reg}$ for classification and regression tasks respectively. In this paper, we used Binary Cross Entropy (BCE) loss to update the network for classification tasks:

$$\mathcal{L}_{cls} = -W\left[y \cdot \log \hat{y} + (1-y) \cdot \log\left(1-\hat{y}\right)\right], \tag{3.9}$$

where W is the weight matrix. And Mean Squared Error (MSE) loss is applied for regression tasks:

$$\mathcal{L}_{reg} = \left(\hat{y} - y\right)^2 \tag{3.10}$$

3.3.2 Gradient Norm

The gradient norm g we used here is not calculated strictly following the common definition of the gradient norm during the network update. It is a relative norm of the input sample's gradient, which reflects if the sample is easy or hard to predict. The term of gradient norm g is used for convenience. Specifically, we measure the distance between the prediction and the ground truth label, and scale the value to (0,1). A similar protocol is also established in [106]. The gradient norm for classification tasks is defined as:

$$g_c = |\hat{y} - y| = \begin{cases} 1-\hat{y} & \text{if } y = 1, \\ \hat{y} & \text{if } y = 0. \end{cases} \tag{3.11}$$

Since $\hat{y}$ in Equation 3.11 represents the predicted probability which is obtained by performing a sigmoid operation on the direct logits output from the prediction network, g_c is capable of indicating how well the sample is predicted. Moreover, we simplified the gradient norm for regression tasks as shown in Equation 3.12, and the value is scaled to (0,1) for better visualization.

$$g_r = (sigmoid|\hat{y} - y| - 0.5) \times 2. \tag{3.12}$$

3.3.3 Attentive Loss

The gradient-norm based attentive loss ($\mathcal{L}_{GNBA}$) is calculated based on the gradient value g. Specifically, a self-attention mechanism is applied on the calculated g during training [82], thus the attention value is learned and updated during the model training by:

$$attn = \mathrm{softmax}\left(W_2 \tanh\left(W_1 g\right)\right), \tag{3.13}$$

where g defers to g_c for classification and g_r for regression, $W_1 \in \mathbb{R}^{d_{\mathrm{attn}} \times 1}$ and $W_2 \in \mathbb{R}^{1 \times d_{\mathrm{attn}}}$ are learnable matrices, d_{attn} is the hidden dimension in the self-attention mechanism. W_1 linearly transforms the gradient norm g to a h_{attn}-dimensional space, while W_2 provides the insights of sample importance, then a softmax function is followed to normalize the importance. Thus, $\mathcal{L}_{GNBA}$ is designed by embedding the attention value $attn$ into the prediction loss for each sample. Suppose the dataset contains molecules $M = \{\mathcal{M}_i\}_{i=1}^{K}$,

$$\mathcal{L}_{GNBA} = \begin{cases} \sum_{\mathcal{M}_i \in M} \mathcal{L}_{cls} \cdot attn & \text{if } classification, \\ \sum_{\mathcal{M}_i \in M} \mathcal{L}_{reg} \cdot attn & \text{if } regression. \end{cases} \tag{3.14}$$

3.4 Experiments and Results

In this section, we describe the implementation of extensive experiments in detail.

Table 3.1: The experiment results on classification and regression datasets respectively. The better score for each method pair is marked as **bold**. Bbbp and bace are used for classification tasks, and the evaluation score is ROC-AUC, higher is better. Lipophilicity and esol are used for regression tasks, and the evaluation score is RMSE, lower is better.

	Classification		Regression	
Method \ Dataset	Bbbp	Bace	Lipophilicity	Esol
GIN	0.920 ±0.011	0.898 ±0.003	0.669 ±0.010	1.074 ±0.112
GIN + attn	**0.937** ±0.002	**0.911** ±0.001	**0.591** ±0.010	**0.956** ±0.027
GAT	0.928 ±0.001	0.905 ±0.003	0.558 ±0.043	0.958 ±0.017
GAT + attn	**0.938** ±0.003	**0.912** ±0.0003	**0.527** ±0.005	**0.891** ±0.014

3.4.1 Experimental Settings

3.4.1.1 Implementation

Our method is implemented on top of the code from [102], which includes the construction of backbone models GIN and GAT. Our experiments are conducted in a pair-wise manner. In specific, we first conduct experiments utilizing GIN and GAT on various datasets, and then we add proposed attentive loss to each model to compare the performance difference.

3.4.1.2 Dataset Split

The dataset is split randomly into train/validation/test with a ratio of 8:1:1, and we ensure the data splits are exactly the same for each pair-wise experiment. All the experiments are run 3 times to alleviate the randomness as well as demonstrate the robustness. We take the average and standard deviation of the evaluation scores as the final results.

3.4.2 Dataset Description and Evaluation

We have conducted extensive experiments for both classification tasks and regression tasks. Bbbp and bace are classification datasets, while lipophilicity and esol are regression datasets.

3.4.2.1 Datasets

bbbp is the Blood-brain barrier penetration dataset [78]. The **bace** dataset is a documentation that records the compounds which may act as the inhibitors of human β-secretase 1 (BACE-1) [83]. The **lipophilicity** dataset is selected from ChEMBL database, which is an important property that influences the molecular membrane permeability and solubility [88]. **Esol** stands for Estimated Solubility, it includes the aqueous solubility information of compounds [87].

3.4.2.2 Evaluation Metrics

In this paper, we use area under the receiver operating characteristic curve (ROC-AUC) as the evaluation criteria for all classification tasks, and root mean square error (RMSE) for all regression tasks.

3.4.2.3 Baselines

The experiments are conducted in a pair-wise manner to verify the effectiveness of the proposed method. Specifically, the baseline model is considered as running with GIN or GAT directly, and our method is implemented by adding the attentive loss to the baseline models. Consequently, as shown in Table 3.1, **GIN** and **GAT** denote the baseline models, while **GIN + attn** and **GAT + attn** represent the proposed method. All the models are run on the four datasets accordingly. Moreover, we keep all the hyper-parameters exactly the same to prove the effectiveness of our method, except for the self-attention mechanism.

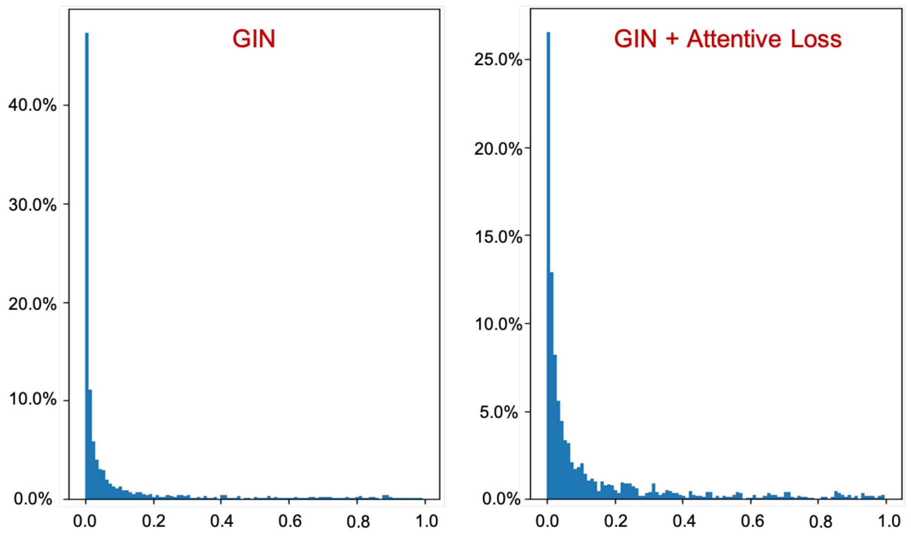


Figure 3.2: An illustration of relative gradient norm distribution between GIN and GIN equipped with attentive loss on bbbp dataset. x-axis represents the values of the gradient norm, and y-axis represents the fraction of the data samples.

3.4.3 Experimental Results

3.4.3.1 Visualization of Gradient Norm Distribution

Based on our assumption, the gradient norm distribution of input dataset should be changed after applying proposed attentive loss. Thus, we first plot the histogram of the data according to the gradient norm after training. Fig. 3.2 demonstrates the distribution of the gradient norm on bbbp dataset. The figure on the left shows the distribution of employing GIN only, while the figure on the right denotes GIN equipped with the self-attention mechanism. As observed, the distribution has changed as expected. For the baseline method, there exists many easy samples with quite small gradients, thus the model training benefits little from them. In the mean time, we observed from our method that the distribution indeed trends to balance the samples by increasing the gradient for those easy samples, which promotes the model to train better.

3.4.3.2 Comparison Results

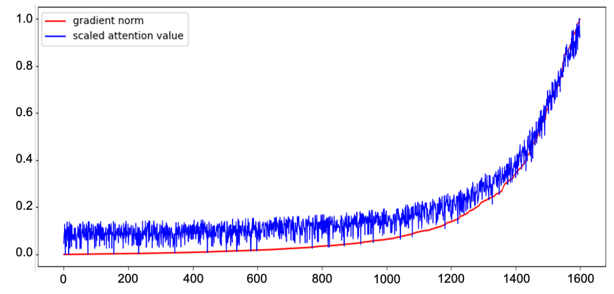
Extensive experiments are conducted on both classification and regression datasets using GIN and GAT as the backbone models. The results are shown in Table 3.1 in a pair-wise manner. We can see that models with proposed attentive loss outperform all the baseline models. The improvement can be up to 1.89% for the classification tasks, and 11.78% for regression tasks. Since our proposed method can be considered as an add-on unit for any prediction model, the influence, in theory, should not be that significant. The self-attention mechanism is able to improve the prediction performance by introducing different weights for each sample according to the prediction difficulty level, while at the same time, not changing the underlying neural network algorithms.

3.4.3.3 Attention Values Visualization

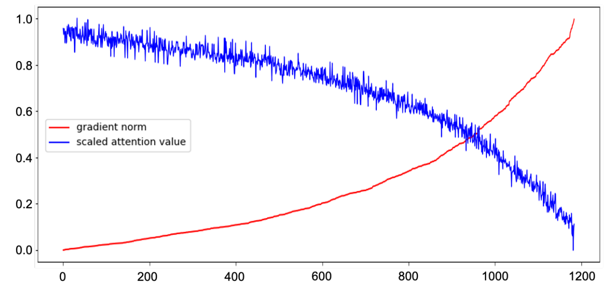
In order to further verify our hypothesis that the network is able to learn the attentions according to the prediction difficulty and may varies from different datasets, we plot the learned attentions along with the gradient norm to check the overall trend between them. Fig. 3.3 demonstrates the gradient norms and the attention values for each dataset, which is generated on the training dataset from the training epoch with the best validation score, with the backbone model of GIN. Since the attention is learned per batch during model training, the value is relatively small compared with the gradient norm. We have applied max-min normalization to scale the attention value to (0,1) for better visualization. The figure is plot by sorting the gradient norms in an ascending order along with the corresponding attention values. As observed, there empirically exists certain relations between the gradient norm and the learned attention. Moreover, it varies for different datasets as expected. For dataset bace, with the increase of gradient norm, the attention is decreased; while for other datasets, both of them follow the same trend. Such observations confirm that the weights associated with each sample shall not be simply defined, it will be related to the data distribution, as well as the prediction difficulty level. For some datasets, paying more attention on the hard samples may help increase the overall prediction performance. However, for some other datasets, those extremely hard samples may be outliers so learning more from them may result in a worse performance for other samples. In consequence, a learnable weight should be adjusted for training. Our proposed method pushes the network to learn how to deal with these samples, and assign an attention value to indicate the importance of each sample. It is empirically demonstrated that our method can boost the training of any backbone models.

3.5 Conclusion

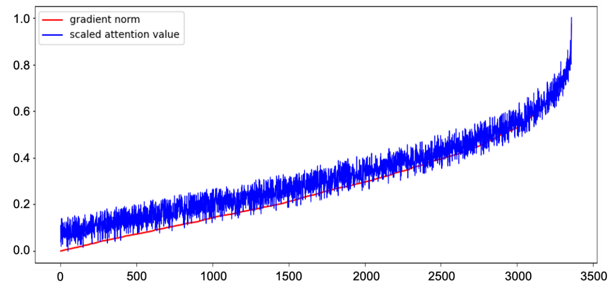
We propose a gradient-norm based attentive loss for molecular property prediction, which is deployed via a self-attention mechanism. Rather than developing training algorithms to improve the prediction performance, we dive into the data level to explore the relationship between each data sample, which brings in a novel perspective to address molecular property prediction in the field. Extensive experiments have confirmed the effectiveness of proposed method. Our attentive loss can also be embedded into any supervised learning models since it only depends on the relative gradient norm. Moreover, the attentive loss is fully data-driven, which means all you need is to equip it with your existing models and the network will do the rest. Our proposed method is the first step towards a data-driven weight learning mechanism to address easy-hard sample imbalance problem in molecular property prediction. Notwithstanding, there still remains unexplored perspectives in such direction, which are our future work. For example, we may perform a case-by-case analysis on the test dataset to see how the predictions change with the learnable weights, or conduct experiments on more datasets to get a comprehensive study across different molecular properties.



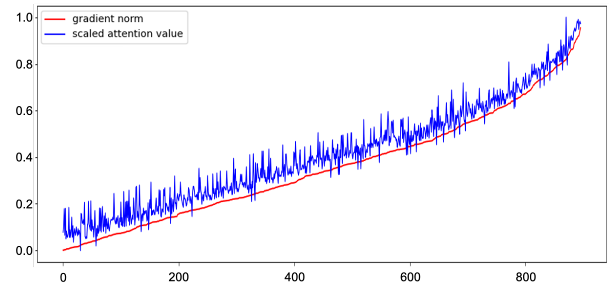
(a) Bbbp.



(b) Bace.



(c) Lipophilicity.



(d) Esol.

Figure 3.3: Visualization of the gradient norms and the scaled attention values for each training dataset with GIN as the backbone model. x-axis denotes each training sample, and y-axis denotes the value of the sample's relative gradient norm and attention. The data is sorted in an ascending order based on the values of relative gradient norm, and the attention values are plotted accordingly.

CHAPTER 4

ROBUST SELF-TRAINING STRATEGY FOR VARIOUS MOLECULAR BIOLOGY PREDICTION TASKS

As discussed before, molecular biology prediction tasks suffer the limited labeled data problem since it normally demands a series of professional experiments to label the target molecule. Self-training is one of the semi-supervised learning paradigms that utilizes both labeled and unlabeled data. It trains a teacher model on labeled data, and uses it to generate pseudo labels for unlabeled data. The labeled and pseudo-labeled data are then combined to train a student model. However, the pseudo labels generated from the teacher model are not sufficiently accurate. Thus, we propose a robust self-training strategy by exploring robust loss function to handle such noisy labels, which is model and task agnostic, and can be easily embedded with any prediction tasks. We have conducted molecular biology prediction tasks to gradually evaluate the performance of the proposed robust self-training strategy. The results demonstrate that the proposed method consistently boosts the prediction performance.

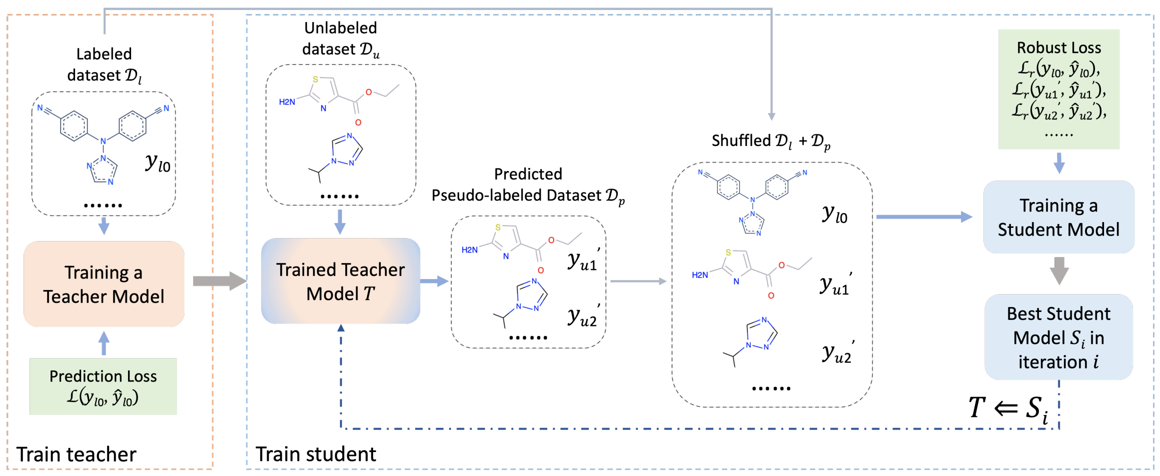
4.1 Introduction

Molecular biology prediction is one crucial and fundamental task for bioinformatics areas such as drug discovery [3,45,109]. It includes various molecule-relevant tasks, such as molecular property prediction and protein secondary or tertiary structure prediction. With the development of deep learning techniques, more and more research tackle these tasks with various deep learning models [1,24,27,40,56,66,101,110]. The prediction task is well-known as a supervised problem, which takes the labeled data as input and employs computational models to predict the corresponding labels. Many existing studies target at such problems in this manner [34, 41, 44, 73, 111]. However, one of the ongoing problems in molecular biology is that labeled data is limited and also difficult to obtain. It usually requires a series of professional experiments, which is time-consuming and costly. Therefore, more paradigms have been developed to utilize unlabeled data to help promote supervised learning, such as semi-supervised learning [99, 102, 112]. Within this field, a simple yet effective paradigm that exploits both unlabeled and labeled data, called self-training, is rarely explored for molecular biology prediction tasks.

In general, self-training is established in four steps: 1) a teacher model is trained on labeled data; 2) the trained teacher model is employed to generate pseudo labels for unlabeled data; 3) the labeled data and the pseudo-labeled data are combined to train a student model; 4) the student model then becomes the teacher model to repeat steps 2-3 until the training is converged. In this fashion, more data is included in the training process, and the student model is able to inherit from the teacher. This paradigm is easy to implement and powerful to boost the training process. Self-training has been widely used in other areas and obtained promising performance, e.g., Computer Vision (CV) [113–115], and Nature Language Processing (NLP) [116–118]. One primary reason is that not only the unlabeled data is enormous, the size of labeled data is also quite large, so are the training models. Thus, the teacher model can sufficiently learn from the labeled data, and achieve favorable performance. Then the student is able to learn better. However, for most molecular biology prediction tasks, the size of the labeled dataset is only a few thousands, and the corresponding prediction performance is not as high as image classification whose accuracy may achieve 95%. Such scenarios lead to a problem: the generated pseudo labels may not be accurate. Such noisy labels may further bias student learning. Therefore, how to handle the label noise is the major concern when establishing self-training strategy in molecular biology area.

One straightforward way to encourage the model to learn from the noisy labels, is to design a loss with regularization to leverage the neural network learning. Mean absolute error (MAE) and cross-entropy (CE) loss are two commonly used loss functions in prediction tasks, where the former is utilized in regression tasks and the latter is used for classification. MAE has been theoretically proved to be robust to label noise during the training, while the CE is not [119]. Recently, robust loss functions have been studied to tackle the noisy label problem in classification tasks by generalizing MAE and CE, and have achieved impressive performance when solving the image classification problem [119–122].

In this paper, we propose to integrate robust loss function and self-training to form a robust self-training framework for molecular biology prediction tasks. Extensive experiments have been conducted over molecular regression and classification tasks to gradually evaluate the effectiveness of proposed robust self-training strategy. Our contributions can be summarized as 1) we are the first to propose a robust self-training paradigm that utilizes robust loss to constrain the student training; 2) the proposed framework is straightforward, and easy to fit into any prediction tasks, which is a simple yet practical strategy to promote the molecular biology prediction tasks; 3) extensive experiments on molecular biology prediction tasks demonstrate that self-training can improve the prediction performance by involving more unlabeled data, and the robust loss can further boost the performance by leveraging the label noise.



Labeled dataset $\mathcal{D}_l$
y_{l0}
Training a Teacher Model
Prediction Loss $\mathcal{L}(y_{l0}, \hat{y}_{l0})$
Train teacher
Unlabeled dataset $\mathcal{D}_u$
Trained Teacher Model T
Predicted Pseudo-labeled Dataset $\mathcal{D}_p$
y_{u1}'
y_{u2}'
Shuffled $\mathcal{D}_l + \mathcal{D}_p$
y_{l0}
y_{u1}'
y_{u2}'
Robust Loss $\mathcal{L}_r(y_{l0}, \hat{y}_{l0})$, $\mathcal{L}_r(y_{u1}', \hat{y}_{u1}')$, $\mathcal{L}_r(y_{u2}', \hat{y}_{u2}')$, ……
Training a Student Model
Best Student Model S_i in iteration i
$T \Leftarrow S_i$
Train student

Figure 4.1: A overview illustration of the robust self-training architecture. More details are described in Section 5.2.

4.2 Methods

4.2.1 Problem Definition

Molecular biology prediction problems can be further referred to as regression problems or classification problems. Given a molecule $\mathcal{M}$, the label needs to be predicted is denoted as y, where $y \in \mathbb{R}$ for a regression problem, and $y \in \{0, 1, \ldots, K-1\}$ for a K-class classification problem. The input molecule $\mathcal{M}$, can be any format according to the task specifics, e.g., protein sequence for protein secondary structure prediction, or molecular graph structure for molecular property prediction. In this study, we conduct two types of experiments to gradually demonstrate the effectiveness of proposed robust self-training strategy: molecular property regression, and molecular property classification.

4.2.2 Robust Self-training Overview

Our proposed robust self-training strategy is implemented on top of the self-training framework. Figure 4.1 illustrates the overall architecture, which can be viewed as two parts, train teacher and train student. First of all, a teacher model is trained on the labeled dataset $\mathcal{D}_l$, and a trained teacher model T is obtained. After that, the student training process begins. It starts with generating the pseudo labels for the unlabeled dataset $\mathcal{D}_u$ to construct a pseudo-labeled dataset $\mathcal{D}_p$. Then the student model is initialized with the teacher model, and trained on shuffled $\mathcal{D}_l + \mathcal{D}_p$. After training for several epochs, we consider that as one iteration, the best model during $i\text{-}th$ iteration is selected as the best student model S_i, then regard it as the new teacher model to repeat the previous steps. This process is repeated for i iterations until the student model is converged.

4.2.3 Molecular Biology Prediction Tasks

Molecular biology prediction task can be considered as two parts in the view of deep learning, which are molecular encoder model and prediction model. Molecular encoder model generates a vector that represents the input molecule, and the prediction model takes the vector to make a prediction. The input molecule can be represented as any format, e.g., sequence or graph structure. We take molecular property regression and classification tasks as examples to evaluate the performance of our proposed strategy. It is noteworthy that any prediction tasks can be adapted with proposed robust self-training since our method generates pseudo labels by training teacher model from the labeled data, such as protein secondary and tertiary structure prediction [27, 123].

In our experiments, we utilize the molecular graph structure and employ two representative graph-based models, EGNN [124] and GIN [56], as the backbone models to predict molecular regression and classification properties. We give a universal definition here for the graph-based encoder and the prediction model.

Molecule $\mathcal{M}$ can be naturally represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $|\mathcal{V}| = p$ refers to the set of p atoms and $|\mathcal{E}| = q$ refers to a set of q bonds in the molecule. The features of atom v is referred as $\mathbf{a}_v \in \mathbb{R}^{d_a}$, and the features of bond (v, u) is referred as $\mathbf{b}_{vu} \in \mathbb{R}^{d_b}$, where $\mathbb{R}^{d_a}$ and $\mathbb{R}^{d_b}$ represent the feature dimension of atom and bond respectively. $\mathcal{N}(v)$ represents the neighbor atoms of atom v, which is identified by the connected bonds. GNN-based models generally perform a message passing and state update protocol for updating atom/bond features. Then, the states of all the atoms are captured to generate a vector representation $\mathbf{h}_{\mathcal{G}}$ through a readout mechanism.

After going through the graph encoder model, the graph representation $\mathbf{h}_{\mathcal{G}}$ is then fed into the prediction model to make a prediction of the property. The prediction model is generally a simple neural network such as multi-layer perceptron (MLP): $\hat{y} = MLP(\mathbf{h}_{\mathcal{G}})$, where $\hat{y}$ is the output of the prediction model, which refers to the predicted probability for the classification tasks or the actual predicted property value for the regression tasks.

Next, each backbone model is introduced along with the employed robust loss respectively.

4.2.3.1 Regression task

As we have mentioned earlier, MAE has been proved to be robust for label noise. Therefore, we first conduct experiments on molecular regression tasks with MAE as the loss function. EGNN [124] is one most recent work to address such problems. Other than the commonly used message passing process based on the graph structure and features, EGNN further explores the geometric information by considering the atom coordinates $\mathbf{x}^d = \left\{\mathbf{x}_0^d, \ldots, \mathbf{x}_{p-1}^d\right\}$. The message update for layer d is defined as:

$$\mathbf{m}_{vu}^d = \phi_e\left(\mathbf{h}_v^d, \mathbf{h}_u^d, \left\|\mathbf{x}_v^d - \mathbf{x}_u^d\right\|^2, e_{vu}\right), \tag{4.1}$$

$$\mathbf{x}_v^{d+1} = \mathbf{x}_v^d + C \sum_{u \neq v} \left(\mathbf{x}_v^d - \mathbf{x}_u^d\right) \phi_x\left(\mathbf{m}_{vu}^d\right), \tag{4.2}$$

Table 4.1: Mean Absolute Error (MAE) for each molecular property regression benchmark on QM9 dataset. Lower is better, best score is marked in **bold**, and green color indicates our proposed method. The last two rows illustrate the improvement percentage by our method compared with others. Avg demonstrates the average score over the row, which denotes the MAE average for all 12 tasks for the first three rows, and the average improvement for the last two rows. Details about each property can be found in [2].

Task	α	$\Delta\varepsilon$	$\varepsilon_{\text{HOMO}}$	$\varepsilon_{\text{LUMO}}$	μ	C_ν	G	H	R^2	U	U_0	ZPVE	Avg
Unit	bohr^3	meV	meV	meV	D	cal/mol K	meV	meV	bohr^3	meV	meV	meV	
EGNN-labeled	0.118	0.070	0.044	0.041	0.057	0.044	0.020	0.021	0.151	0.020	0.019	2.111	0.226
EGNN-self-training (Ours)	**0.067**	**0.048**	**0.028**	**0.025**	**0.028**	**0.031**	**0.010**	**0.010**	**0.083**	**0.011**	**0.010**	**1.521**	**0.156**
EGNN-all	0.071	**0.048**	0.029	**0.025**	0.029	**0.031**	0.012	0.012	0.106	0.012	0.011	1.55	0.161
Ours v.s. EGNN-labeled ↑	+43.2%	+31.6%	+36.1%	+38.4%	+50.6%	+30.0%	+50.7%	+52.8%	+45.2%	+43.6%	+49.2%	+28.0%	+41.5%
Ours v.s. EGNN-all ↑	+5.6%	0.00%	+3.5%	+0.00%	+3.5%	+0.00%	+16.7%	+16.7%	+21.7%	+8.3%	+9.1%	+1.9%	+7.2%

$$\mathbf{h}_{\mathcal{N}(v)}^{d+1} = \text{AGGREGATE } \left(\left\{\mathbf{m}_{vu}^{d}, \forall u \in \mathcal{N}(v)\right\}\right), \tag{4.3}$$

where x_v^d and x_u^d are the coordinates of atom v and its neighbor atom u at *d-th* step, vu represents the bond between them, e_{vu} denotes the bond features, ϕ_e and ϕ_x are two output operations, and C equals $1/(p-1)$.

MAE is used as the robust loss function to constrain the network training with regards to noisy labels, which is defined as:

$$\mathcal{L}_{MAE} = \frac{1}{M}\sum_{i=1}^{M} \left| y_i - \hat{y}_i \right|, \tag{4.4}$$

where M is the size of the dataset.

4.2.3.2 Classification task

Classification tasks are dominating for molecular biology prediction problems as well. We then conduct experiments over molecular property classification tasks to evaluate the effectiveness of the self-training paradigm. However, the commonly used cross-entropy (CE) loss is not robust, so we employ the generalized cross-entropy (GCE) loss [120] to boost the self-training. The backbone model utilized for this task is GIN [56]. GIN is theoretically proved as one of the most powerful GNN models. It utilizes multi-layer perceptron (MLP) for state update, and employs a concatenate operation over all passing steps during the readout phase. The updated rule can be summarized as:

$$\mathbf{h}_v^{d+1} = \mathrm{MLP}^{d+1}\left(\left(1+\epsilon^{d+1}\right)\cdot \mathbf{h}_v^d + \sum_{u\in\mathcal{N}(v)} \mathbf{h}_u^d\right), \tag{4.5}$$

$$\mathbf{h}_{\mathcal{G}} = \mathrm{CONCAT}\left(\mathrm{READOUT}\left(\left\{\mathbf{h}_v^{d+1} \mid v \in \mathcal{V}\right\}\right)\right), \tag{4.6}$$

where ϵ is a fixed scalar or a learnable parameter.

GCE loss is a generalized version of CE and MAE. The CE loss is defined by:

$$\mathcal{L}_{CE} = -\sum_{k=1}^{K} y^k \log \hat{y}^k, \tag{4.7}$$

for a K-class classification problem (K=2 for binary classification), where y^k is the one-hot encoding label, and $\hat{y}^k$ denotes the probability output from the prediction network. Allow $f_k(x) = \hat{y}^k$, GCE loss is designed by:

$$\mathcal{L}_{GCE} = \frac{(1 - f_k(x)^q)}{q}, \text{ where } q \in (0,1]. \tag{4.8}$$

GCE loss is reduced to CE loss and MAE loss when $q \to 0$ and $q = 1$, respectively. Detailed proofs can be found in [120].

4.3 Experiments

Extensive experiments are conducted gradually to evaluate the performance of proposed robust self-training strategy. Since MAE is theoretically proved to be robust to label noise, we first implement self-training on molecular regression task, and utilize MAE loss as the robust loss function to demonstrate the superiority of proposed method. Then we explore GCE loss on the molecular classification task to further confirm the effectiveness of integrating robust loss function with self-training.

4.3.1 Datasets Description and Setup

QM9 [2] is a standard benchmark for molecular property regression problem. It is a subset of GDB-17 database [125], which contains 134k molecules. It comprehensively provides 12 quantum chemical properties for each molecule, including geometric, energetic, electronic, thermodynamic, etc. **HIV** is introduced by the Drug Therapeutics Program (DTP) AIDS Antiviral Screen. It contains the test result of 41,127 molecule compounds with the ability for inhibiting HIV replication. The widely used version provided by MoleculeNet contains inactive labels and activa labels, which makes it a binary classification task [66].

For all tasks, we randomly select 50% of the data as the unlabeled dataset, and the rest is used as the the labeled dataset with a 3:1:1 training/validation/test ratio. We do not use an external unlabeled dataset here since most molecules may not express target property at all, which may lead to a biased comparison.

4.3.2 Experimental Details

4.3.2.1 Baselines

For all the experiments, we consider training solely on the labeled datasets as the fundamental baselines, which is denoted as ”-labeled”. Then, we establish our vanilla implementation by running experiments with self-training paradigm on both labeled dataset and unlabeled dataset, denoted as ”-self-training”. Last, we integrate robust loss with our vanilla self-training benchmark to demonstrate the superiority of our robust self-training, denoted as ”-robust”. Since the unlabeled dataset is formed by randomly selecting 50% from the original labeled dataset, we also compare the performance when using the original backbone model without self-training on all the data with labels, denoted as ”-all”.

4.3.2.2 Configurations

We follow the original implementation and settings of the backbone models, and implement robust self-training on top of them. All the hyper-parameters of the backbone models remain the same to ensure a fair comparison. For the settings of robust self-training, we perform three iterations for the student training, and tune the hyper-parameter q when employing GCE loss. For molecular classification task, we run the experiments three times to alleviate the randomness since HIV dataset is much smaller than other datasets, leading to relatively unstable performance. We take the average and standard deviation of the evaluation scores as the final results. For molecular regression task, we follow the original configurations and evaluations to run the experiments one time. The results do not vary much since the training data is sufficiently large and the converged stage is stable.

4.3.2.3 Training strategy

We follow the same procedure for all three tasks. First, we train a teacher model on the labeled data, and use it to generate pseudo labels for the unlabeled dataset. Next, for the vanilla self-training, we train the student model which takes the teacher model as the initialization on the combined labeled and pseudo-labeled dataset. Note that the pseudo-labeled dataset is only merged into the training dataset along with the labeled training dataset. The validation and test datasets remain the same from the teacher model training. Furthermore, we choose the best student model in the current iteration as the new teacher model to generate a new pseudo-labeled dataset and initialize the student model for the next iteration. We run the student training for three iterations, and take the best validation model to evaluate the test dataset performance. For robust self-training, the procedure is the same as vanilla self-training, except robust loss function is employed.

4.3.3 Experimental Results

Our first experiment is to employ the self-training paradigm directly on molecular regression tasks, since MAE is theoretically robust to noisy labels. The comparison results for each property are shown in Table 4.1. As we can observe, the performance of the self-training strategy outperforms EGNN-label consistently by a 41.5% average improvement. Moreover, the performance is competitive against the supervised training on the all-labeled dataset. Our implementation achieves the best performance on 9/12 tasks compared with the original EGNN-all on all 134k labeled data, which gains the average MAE boost by 7.2%. The experiments on regression tasks with MAE sufficiently demonstrate that robust loss function is a perfect fit for self-training strategy by dealing with the generated pseudo labels.

Table 4.2: ROC-AUC score for molecular property classification benchmark on HIV dataset. Higher is better, best score is marked in **bold**, and green color indicates our proposed method.

	GIN-labeled	GIN-self-training	GIN-robust	GIN-all
HIV	$0.786_{\pm 0.008}$	$0.798_{\pm 0.005}$	$\mathbf{0.822}_{\pm 0.005}$	$0.820_{\pm 0.015}$

We then conduct experiments on the HIV dataset to evaluate how robust self-training performs on classification task. As shown in Table 4.2, the improvement of directly implementing self-training is limited, which is reasonable since CE loss is not theoretically robust [119]. Therefore, we explore robust loss function GCE and integrate it with self-training to form the robust self-training paradigm, which further boosting the ROC-AUC to 0.822. Moreover, our method is competitive with the original GIN implementation on the all-labeled dataset with a 0.002 improvement. Note that in our self-training experiments, 50% of the dataset is treated as unlabeled, while GIN-all is trained on 100% labeled dataset.

Extensive experiments empirically demonstrate that our proposed robust self-training strategy is capable of efficiently exploring both labeled and unlabeled data as well as handling the noisy pseudo labels. Moreover, we roughly conduct experiments on protein secondary structure prediction task by adopting GCE loss, and has achieved promising results, which further proves the effectiveness of our proposed strategy. Next, we will explore more details regarding the robust loss type with different prediction tasks.

4.4 Conclusion

In this study, we propose a robust self-training paradigm for various molecular biology prediction tasks by exploring robust loss function to constrain the self-training process. We first train a teacher model on labeled dataset, then use the teacher model to generate pseudo labels for the unlabeled dataset. Next, the student model is trained on the combination of the labeled dataset and the pseudo-labeled dataset. This process is iterated by regarding the student as the new teacher and re-generating the pseudo-labeled dataset until the training is eventually converged. Since the pseudo labels are not the ground-truth labels which means noises exist, we then utilize robust loss function to restrain the student training. Extensive experiments have demonstrated that self-training accompanied with robust loss can boost the prediction performance by taking advantage of both labeled and unlabeled data. Moreover, our proposed robust self-training is model and task agnostic, which can be easily inserted into any molecular biology prediction tasks, and benefits the general computational molecular biology society.

CHAPTER 5

DEEP GRAPH LEARNING WITH PROPERTY AUGMENTATION FOR PREDICTING DRUG-INDUCED LIVER INJURY

This chapter addresses one real-world limited data application, Drug-induced liver injury (DILI) prediction. DILI prediction is one of the most challenging and critical tasks in drug discovery since DILI is toxic for human beings. DILI is a crucial factor in determining the qualification of potential drugs. However, the DILI property is excessively difficult to obtain due to the complex testing process. Consequently, an *in silico* screening in the early stage of drug discovery would help to reduce the total development cost by filtering those drug candidates with high risk to cause DILI. To serve the screening goal, we apply several computational techniques to predict DILI property, including traditional machine learning methods and graph-based deep learning techniques. While deep learning models require large training data to tune huge model parameters, the DILI dataset only contains a few hundreds of annotated molecules. To alleviate the data scarcity problem, we propose a property augmentation strategy to include massive training data with other property information. Extensive experiments demonstrate that our proposed method significantly outperforms all existing baselines on DILI dataset by obtaining a 81.4% accuracy using cross-validation with random splitting, 78.7% using leave-one-out cross-validation, and 76.5% using cross-validation with scaffold splitting.

5.1 Introduction

Drug discovery has been a critical research area for years. The development process of new drugs is extremely time consuming and resource costly since it usually requires a series of complicated *in vitro* and *in vivo* experiments [3, 126, 127]. One major challenge is to identify the safety of the potential drug candidates, e.g. filtering the drugs that may cause human toxicity. Drug-induced liver injury (DILI) is one of the most fundamental toxicity concerns that are undesirable and unpredictable. Several research indicate that traditional hepatotoxicity testings on animal models may have distinct outcomes from humans [128–130]. Since animal or human model testings are usually conducted in the late stage of drug development, the withdrawal or termination of such disqualified drug candidates would sacrifice lots of previous efforts. Therefore, a precise and accurate model to better predict DILI in the early stage would be a promising approach to facilitate the development progress.

Human toxicity data is extremely hard to collect, since *in vivo* and *in vitro* toxicological studies cannot provide adequate assessment when the drug candidates are applied on human [128–131]. Several labeling schemes [132–135] have been developed to annotate DILI label for certain drugs to provide predictive models with labeled data. [135] is based on physician desk reference, while others [132–134] are coming from case reports and literature. Although labeled DILI datasets are available in public, such datasets only contain one or two hundreds of drugs, and what is worse, the labeling standards are inconsistent. To tackle this problem, FDA develops an annotation scheme to label DILI risk of 1036 FDA-approved drugs, and announces the DILIrank [136] dataset in 2016. The previous version of DILIrank annotates the drugs with Most-DILI concern, Less-DILI concern, and No-DILI concern, based on the regulatory professionals assessment [137]. The new scheme establishes a more detailed verification process dividing the drugs into four categories: Most-DILI concern, Less-DILI concer, No-DILI concern, and Ambiguous DILI concern [136]. DILIrank is the most widely used dataset to develop predictive models of DILI, and has been used in various studies [138–141]. Lately, FDA further augments DILIrank to DILIst [128] with other four literature datasets by applying concordance analysis across these five datsets. Until now, DILIst is the largest dataset with DILI classification, which contains 1279 drugs. These efforts [128, 136] provide invaluable resource for predicting DILI risk.

DILI prediction can be considered as the application of molecular property prediction, which is one of the oldest cheminformatics tasks. Many *in silico* methods have been applied to solve molecular property prediction problem [34, 60, 61, 142]. These approaches generally convert the molecule into a vector representation via different procedures, and then go through different machine learning models to predict the label information. The vector representation of a molecule is called fingerprints. Traditionally, fingerprints are either manually constructed by experts (hand-crafted biologist-guided fingerprints), or calculated by a fixed hash function (hash-based fingerprints). The former one is designed by specialists based on biological experiments and chemical knowledge. Specific sub-structures of the compounds are considered as functional groups, and their corresponding local features are determined based on their properties revealed during experiments or different states [60, 142]. E.g., $CC(OH)CC$ appears to have solubility relevant characteristic; thus it has been isolated as local features to produce fingerprints on solubility related tasks. Hash-based fingerprints such as circular fingerprints employ a fixed hash function to extract each layer's feature of a molecule based on the concatenated features of the neighborhood in the previous layer [61]. This type of the fingerprints is non-invertible, so there is no way to check back and modify the quality of the fingerprints if the hash function cannot capture enough information, which might lead to poor performance in further predictive tasks. To tackle this problem, [143] recently proposes a reverse-engineering method to reconstruct the molecular structure from hash-based fingerprints such as ECFP [61].

With the rapid increase of deep learning techniques, recent studies trend to address molecular property prediction with such novel models. One promising research interest is considering a molecule as a graph, since the atoms of the molecules can be referred as the vertexes, and the bonds between atoms as the edges. Neural fingerprints [34] are the first attempt to learn molecular vector representation based on its graph structure. The difference between neural fingerprints and hash-based fingerprints is the replacement of the hash function. Neural fingerprints apply a non-linear activated densely connected layer to generate the fingerprints. Many other graph-based deep learning models can also be applied to represent a molecule by embedding the graph features to a continuous vector [35, 66]. Within them, the Message Passing Neural Networks (MPNN) [1,73] have achieved notable prediction performance. MPNN models recursively update the atom or bond features by aggregating message/information from its adjacent atoms or bonds, then employ a readout function to pool all updated features of atoms to deliver the global representation of the molecule. However, these methods only focus on one single view of the graph topology, either atom-central or bond-central. Taking Figure 5.1 as an example, the left graph is the atom-oriented structure of caffeine, and the right one is its bond-oriented representation. It is observed that both atom and bond features should be taken into account when embedding a molecule graph, e.g., the double bond within the benzene $N = C$ is distinct from bond $C = O$, atom N and C are notably different. Inspired by this insight, we propose a fresh perspective of viewing the graph from two aspects in our recent work $\mathsf{MV-GNN}^{\mathsf{cross}}$ [144], which involves both atom messages and bond messages. $\mathsf{MV-GNN}^{\mathsf{cross}}$ model takes the molecular SMILES as input, and use RDKit [80] to extract the graph

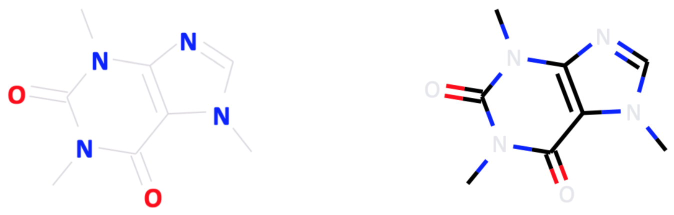


Figure 5.1: Atom-oriented graph v.s. Bond-oriented graph.

structure and the local features associated with each atom and bond. A graph encoder network then learns and converts such information into a vector representation of the input molecular SMILES. After that, the vector representation is fed into a prediction network to predict the property label. Our method outperforms all previous SOTAs on 11 commonly used molecular property prediction tasks. Therefore, we employ our graph-based deep learning model on DILIrank dataset to classify the DILI label, and have achieved superior prediction performance compared with other models including both graph-based deep learning models and traditional fingerprints-based models.

Other than that, available labeled DILI drugs are still quite limited for data-hungry deep learning models. In order to get better and more stable prediction performance, several research have been done from different aspects. [128] develops a new annotation scheme to augment the drug list with DILI risk. [138] employs different machine learning models on different human toxicity dataset to investigate the corresponding prediction performance. [141] and [140] propose to obtain better prediction results with ensemble computational models and various molecular descriptors. These attempts have earned certain achievement, but may still be restricted by the available labeled DILI data. To tackle this bottleneck and reinforce the expressive power of deep learning model, we propose a property augmentation strategy to utilize $\mathsf{MV}-\mathsf{GNN}^{\mathsf{cross}}$ models along with more data by taking advantage of other property information. In particular, we create a larger training dataset by combining more drugs with other toxic properties, such as PLD [145] which measures the organism-level toxicity of compounds. Since graph neural network is able to learn molecular vector representation only based on its graph structure and the underlying atom/bond level features, more input data shall help generate more accurate molecular representation. Moreover, for those properties with more available data, deep learning techniques are more likely to obtain better performance. Thus, the correct prediction would help promote the entire training including those properties with only few samples, such as DILI. In this fashion, we are able to increase the accuracy of DILI to 81.4% using cross-validation with random splitting, 78.7% using leave-one-out cross-validation, and 76.5% using cross-validation with scaffold splitting, which is regarded as the remarkable boost considering the challenges on DILI risk prediction. Detailed methodologies and experimental procedures are described in later sections.

5.2 Methodologies

We take our recent work $\mathsf{MV-GNN}^{\mathsf{cross}}$ model as the backbone to implement proposed property augmentation method, since $\mathsf{MV-GNN}^{\mathsf{cross}}$ outperforms other baseline models on DILI dataset in extensive experiments. As shown in Figure 5.2, $\mathsf{MV-GNN}^{\mathsf{cross}}$ contains two principal parts, the **Encoder Network** and the **Prediction Network**. The Encoder Network transforms the input molecular SMILES into a vector representation based on its graph structure, and the Prediction Network is responsible for classifying the binary label of certain properties, such as DILI. Beyond that, we employ deep multi-label learning to establish proposed method while involving more properties information along with DILI.

5.2.1 Molecular Graph Preliminaries

A molecule can be naturally represented as a graph based on its chemical structure, in particular, by taking the atoms as the nodes, and the bonds between

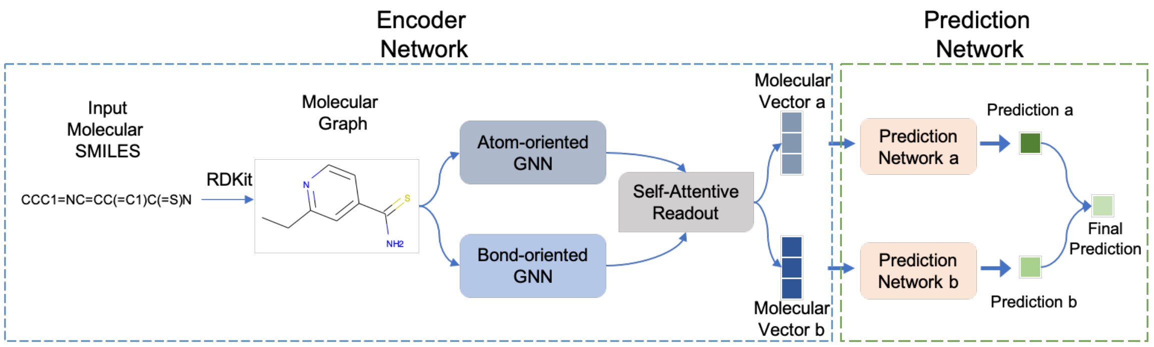


Figure 5.2: Overview of $\mathsf{MV-GNN}^{\mathsf{cross}}$ models.

atoms as the edges. Thus, the molecular graph is denoted as $G_m = (\mathcal{A}, \mathcal{B})$, where $\mathcal{A}$ is a set of the atoms, and $\mathcal{B}$ is a set of the bonds. Based on such graph structure, the initial features of atoms and bonds are extracted as the learning information, and referred as x_a and y_b. Figure 5.4 takes ethionamide as an example to illustrate how a molecule converts to its corresponding computational graph.

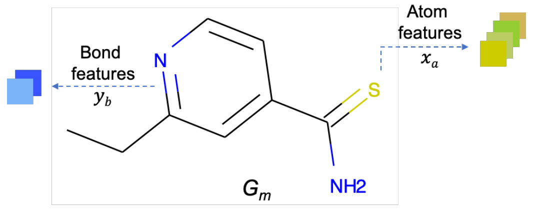


Figure 5.3: Graph definition of ethionamide. G_m represents the entire graph structure, x_a and y_b refer to the atom and bond features that associates with each atom and bond, respectively.

The initial features for each atom and bond is selected follow the same protocol of [1], as shown in Table 5.1 and Table 5.2. All the features are one-hot encodings except the atomic mass, and are extracted using RDKit [80].

Table 5.1: Atom features selection [1].

Features	Size	Descriptions
atom type	100	type of atom (e.g., C, N, O), in the order of atomic number
formal charge	5	integer electronic charge assigned to atom
number of bonds	6	number of bonds the atom is connected
chirality	4	Unspecified, tetrahedral CW/CCW, or other.
number of Hs	5	number of bonded hydrogen atoms
atomic mass	1	mass of the atom, divided by 100
aromaticity	1	whether this atom is part of an aromatic system
hybridization	5	sp, sp2, sp3, sp3d, or sp3d2

Table 5.2: Bond features selection [1].

Features	Size	Descriptions
bond type	4	single, double, triple, or aromatic
stereo	6	E/Z, cis/trans, any, or none
in ring	1	whether the bond is part of a ring
conjugated	1	whether the bond is conjugated

5.2.2 Encoder Network

Molecules can be observed from two perspectives, one is that taking the atoms as the centers and bonds as the connections [73], while the other one is to consider bonds as the centers and atoms as connections [1]. Inspired by multi-view learning [146], $\mathsf{MV-GNN}^{\mathsf{cross}}$ takes advantage of the two perspectives, and design a multi-view framework to generate more informative molecular representation. In specific, the encoder network is constructed by two streams, atom-oriented and bond-oriented, where each contains one Graph Neural Network (GNN). Next, a self-attentive readout mechanism is employed to convert the learned molecular feature matrix to a vector representation.

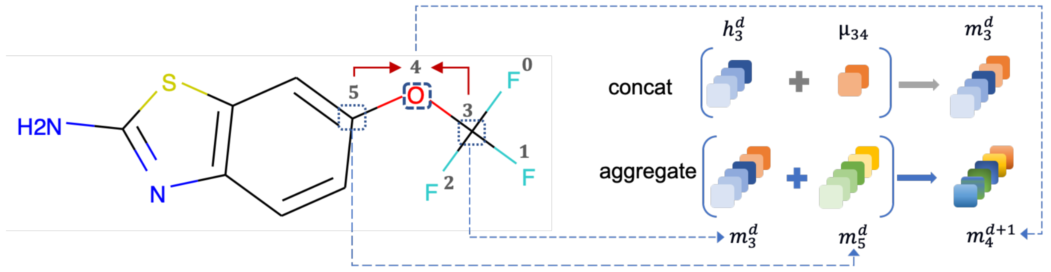


Figure 5.4: Message passing aggregation phase. Taking atom 4 as an example, atom 3 and atom 5 are its neighbors. In the passing process, the message of atom 3 and atom 5 from previous passing step will be aggregated to atom 4. For the message construction, we take atom 3 as an example. The message m_3^d of atom 3 is concatenated by the initial atom features h_3^d of atom 3, as well as the initial bond features μ_{34} of the connected bond 34.

5.2.2.1 Atom-oriented GNN and Bond-oriented GNN

The **Atom-oriented GNN** learns the molecular representation by aggregating neighbor atoms recursively for several steps, while **Bond-oriented GNN** establishes similar procedure via a bond-central fashion. The generalized GNN can be defined as:

$$\begin{aligned} m_o^{d+1} &= \sum_{\eta \in \mathcal{N}(o)} \mathscr{A}_d \left(h_\eta^d, \mu_{\text{attached}} \right) \\ h_o^{d+1} &= \mathscr{U}_d \left(h_o^d, m_o^{d+1} \right). \end{aligned} \tag{5.1}$$

In (5.1), $\mathscr{A}_d$ and $\mathscr{U}_d$ represent the neighbor aggregation function and state update function respectively. m_o^{d+1} and h_o^{d+1} are the aggregated message and states vector for entity o at $d+1$ step respectively. Entity o can be either atoms or bonds. $\mathcal{N}(o)$ is the neighborhood entity set of entity o. μ_{attached} is the attached features of entity o during aggregation. In Atom-oriented GNN, entity o represents the atoms, μ_{attached} denotes the features for the connected bonds. The Bond-oriented GNN is formed with a similar implementation by considering the bonds as passing centers, and atom features as attached. Specially, entity o represents the bonds, and the corresponding bond messages m_o^{d+1} are constructed by bond states vector h_o^{d+1} and attached atom features μ_{attached}.

5.2.2.2 Self-Attentive Readout

The outputs of the two GNN models are the learned feature matrices by regarding molecular graph as atom-oriented and bond-oriented. As demonstrated in Figure 5.2, in order to obtain the fixed length of molecular vector representation, a readout transformation is need to eliminate the obstacle of size variance and permutation variance. Other than commonly used mean-pooling or max-pooling, a self-attentive readout is employed here to generate molecular representation associated with different attention weights [46, 82]. Formally, take a output of Atom-oriented GNN $\boldsymbol{H}_n$ as an example, the self-attention over atoms is defined as:

$$\boldsymbol{S} = \operatorname{softmax}\left(\boldsymbol{W}_2 \tanh\left(\boldsymbol{W}_1 \boldsymbol{H}_n\right)\right), \quad \boldsymbol{\xi}_n = \mathsf{Flatten}(\boldsymbol{S}\boldsymbol{H}_n^\top), \tag{5.2}$$

where n is the number of atoms in the molecule. $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are learnable matrices, which are shared between the two streams to enable message circulation during the multi-view training process. Thus, two molecular vectors are generated in a multi-view manner.

5.2.3 Prediction Network

In $\mathsf{MV-GNN}^{\mathsf{cross}}$, we have generated two vectors from the two sub-modules: atom-oriented GNN and bond-oriented GNN. These two vectors are fed into two prediction networks to make the predictions. Since the two vectors generated via atom-oriented GNN and bond-oriented GNN are coming from the same input SMILES, so the predictions should be the same. Thus, we employ MSE loss to restrain the training, called disagreement loss. Formally, we formulate this molecular property prediction loss as follows:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{pred}} + \lambda \mathcal{L}_{\text{dis}}, \tag{5.3}$$

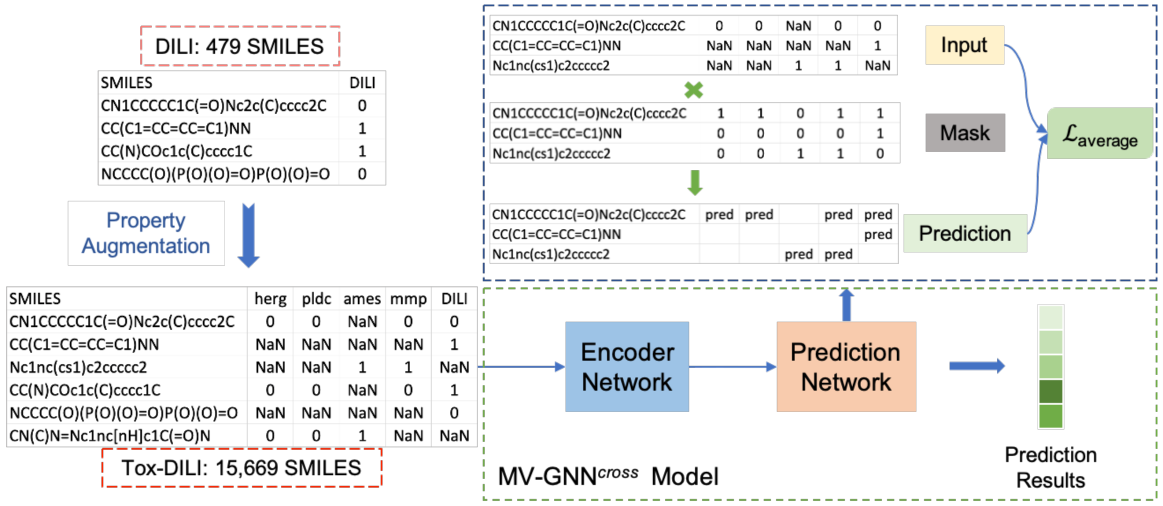
where $\mathcal{L}_{\text{pred}}$ is the supervised loss for each prediction and $\mathcal{L}_{\text{dis}}$ is the disagreement loss between two classifiers.

5.2.4 Property Augmentation Learning

DILI dataset only contains a few hundreds of drugs, which is extremely small for deep learning. In order to take advantage of the expressive power of deep graph learning models such as $\mathsf{MV-GNN}^{\mathsf{cross}}$, we demand more information to boost the training. Since DILI is a property of human toxicity, we compare it with other four available human toxicity datasets: herg [147, 148], PLD [145], ames [149, 150], and mmp [151, 152]. We notice there are overlapping molecules between DILI and these four toxicity datasets. We assume that such correlation may help the training of DILI. Hence, we propose to utilize these additional toxicity information to promote the prediction performance of DILI.

5.2.4.1 Multi-label Training

As shown in Figure 5.5, original DILI dataset contains only 479 SMILES. We take it with other four toxicity properties (herg, PLD, ames, and mmp) which are provided by NIH, to form a larger dataset. Specifically, we combine these five datasets based on the SMILES representation of the drugs. Thus, a large matrix contains 15,669 data samples is generated, where each row stands for one SMILES, and the five columns are the corresponding property labels. Each SMILES could have one or more property labels, and those properties which are not observed for each SMILES are



SMILES	DILI
CN1CCCCC1C(=O)Nc2c(C)cccc2C	0
CC(C1=CC=CC=C1)NN	1
CC(N)COc1c(C)cccc1C	1
NCCCC(O)(P(O)(O)=O)P(O)(O)=O	0

SMILES	herg	pldc	ames	mmp	DILI
CN1CCCCC1C(=O)Nc2c(C)cccc2C	0	0	NaN	0	0
CC(C1=CC=CC=C1)NN	NaN	NaN	NaN	NaN	1
Nc1nc(cs1)c2ccccc2	NaN	NaN	1	1	NaN
CC(N)COc1c(C)cccc1C	0	0	NaN	0	1
NCCCC(O)(P(O)(O)=O)P(O)(O)=O	NaN	NaN	NaN	NaN	0
CN(C)N=Nc1nc[nH]c1C(=O)N	0	0	1	NaN	NaN

CN1CCCCC1C(=O)Nc2c(C)cccc2C	0	0	NaN	0	0
CC(C1=CC=CC=C1)NN	NaN	NaN	NaN	NaN	1
Nc1nc(cs1)c2ccccc2	NaN	NaN	1	1	NaN

CN1CCCCC1C(=O)Nc2c(C)cccc2C	1	1	0	1	1
CC(C1=CC=CC=C1)NN	0	0	0	0	1
Nc1nc(cs1)c2ccccc2	0	0	1	1	0

CN1CCCCC1C(=O)Nc2c(C)cccc2C	pred	pred		pred	pred
CC(C1=CC=CC=C1)NN					pred
Nc1nc(cs1)c2ccccc2			pred	pred	

Figure 5.5: Property augmentation procedure. Original DILI dataset is augmented to Tox-DILI dataset. Tox-DILI is then fed into $\mathsf{MV-GNN}^{\mathsf{cross}}$ model for prediction. During the training period of the prediction network, a mask scheme is applied to handle the back-propagate of missing labels, and an average loss across all properties is used to restrain the entire training.

marked as missing values, and are represented as NaN. The constructed Tox-DILI then goes through $\mathsf{MV-GNN}^{\mathsf{cross}}$ model to classify the labels. We employ a multi-label training approach to establish the property augmentation learning process. During the training process, all property predictions share the same encoder network, and make prediction for each property label individually. Then, the average of all the prediction loss is used to update the neural network parameters. We treat each property equally important, and ignore the prediction for those NaN properties to avoid deviation.

5.2.4.2 Missing Labels Handling

In order to eliminate the effects of the missing labels during the training period, we need to identify such labels for each SMILES, and ignore them during the back-propagation. In our experiments, a mask scheme is implemented as the filter. The mask is a matrix with exact same size of the input, which is applied in the prediction network. While the prediction is made by the prediction network, and the loss is calculated for each data sample, the mask is then multiplied with the loss values. The mask matrix is filled by 0s and 1s, as the corresponding positions with missing labels are recorded as 0, others as 1. Thus, any weights associated with those missing labels would have no influence on the further computation.

Since each SMILES may have multiple binary property labels at the same time, such task could be regarded as multiple binary classification problem. Hence, we employ the Binary Cross Entropy (BCE) loss as the prediction loss function, and compute the average loss across each property. Suppose the dataset contains molecules $\mathcal{M} = \{M_i\}_{i=1}^{K}$, formally, we formulate the final loss processed by the mask as follows:

$$\mathcal{L}_{\text{pred}} = \frac{1}{N * K} \sum_{n=1}^{N} \sum_{M_i \in \mathcal{M}} (\mathcal{L}_{\text{a}}(y_i, \gamma_{a,M_i}) * mask + \mathcal{L}_{\text{b}}(y_i, \gamma_{b,M_i} *) * mask), \tag{5.4}$$

where γ_{a,M_i} and γ_{b,M_i} are the output predictions produced by the two prediction networks, $\mathcal{L}_{\text{a}}$ and $\mathcal{L}_{\text{b}}$ are the corresponding computed loss. y_i is the ground truth label, and N is the total number of properties, which is 5 in our experiments here.

5.2.5 Evaluation Criteria

Since our task is to predict the binary label of DILI by considering Most-DILI-Concern as the positive label and No-DILI-Concern as the negative label, we thoroughly evaluate the performance of each method by calculating the *accuracy*, *sensitivity*, *specificity*, *F1-score*, *Matthews correlation coefficient* and *ROC-AUC*. The accuracy score is the total percentage of the correct predictions of DILI label. Sensitivity is also called true positive rate, which measures the percentage that drugs with positive DILI labels are truly predicted as positive. Specificity is the true negative rate, which represents the rate that drugs without DILI risks are correctly predicted as negative labels. F1-score is the weighted average of precision and recall, where precision is the ratio of the correct positive predictions to all positive predictions, and recall is the ratio of the correct positive predictions to all ground truth positive labels. Matthews correlation coefficient (MCC) leverage the performance of all of the four confusion matrix categories (true positives, false negatives, true negatives, and false positives). ROC-AUC measures the separability of the model to correctly predict positive labels as positive, and negative labels of negative. In addition, we evaluate statistical significance using one-sided Wilcoxon signed-rank test.

5.3 Experiments

We have conducted extensive experiments using Circular-fp [61], Neural-fp [34], MPNN [73], DMPNN [1], and $\mathsf{MV-GNN}^{\mathsf{cross}}$ [144] on DILI to validate the performance. Beyond that, we take $\mathsf{MV-GNN}^{\mathsf{cross}}$as backbone, and employ our proposed property augmentation approach to involve more data, in order to further boost the prediction performance of DILI. Moreover, we conduct additional experiments using MPNN and DMPNN on augmented Tox-DILI dataset to proof the effectiveness of our method.

5.3.1 Dataset Description

Two datasets are used during the experiments, DILI and Tox-DILI[1]. **DILI** is the DILI dataset provided by NIH, which contains 479 molecules with DILI label. The original DILI dataset is coming from DILIrank [136] dataset, which contains 197 molecules with **Most-DILI-Concern**, 282 molecules with **No-DILI-concern**, and 464 molecules with **Less-DILI-Concern**. We consider Most-DILI-Concern as label 1, and No-DILI-concern as label 0 to solve the classification problem. Thus, 479 molecules in total are selected to constitute DILI dataset. The **Tox-DILI** is formed

[1]Refer to supporting information.

by DILI and other four datasets with toxicity relevant properties: herg [147, 148], PLD [145], ames [149, 150], and mmp [151, 152]. The description of each property is stated in Table 5.3, and the label distribution is shown in Table 5.4.

Table 5.3: Description of four toxicity properties used for augmentation.

Category	Property	Description
Toxicity	herg [147, 148]	measures cardiotoxic effects of compounds.
	PLD [145]	stands for phospholipidosis, which measures organism-level toxicity of compounds.
	ames [149, 150]	measures mutagenicity, one of the most important end points of toxicity.
	mmp [151, 152]	The mitochondrial membrane potential (MMP) is a key parameter for evaluating mitochondrial function.

Table 5.4: Datasets statstics.

Dataset	Dataset Size	Property	# Molecules	# Label 0	# Label 1
DILI	479	DILI	479	282	197
Tox-DILI	15,669	herg	3,024	2,541	483
		PLD	4,159	3,777	382
		ames	7,940	4,534	3,406
		mmp	5,970	5,070	900
		DILI	479	282	197

5.3.2 Comparison Experiments

5.3.2.0.1 Circular-fp. Circular fingerprints (Circular-fp) is one of the traditional ways to generate a so-called fingerprints to represent the molecule. It is a vector representation that generated by a hand-crafted hash-based algorithm to define the local features. Circular-fp employs a fixed hash function to extract each layer's features of a molecule and concatenate them together. The generated vector representations usually go through machine learning models to perform further predictions, we apply GradientBoost [65] model here in the experiments.

5.3.2.0.2 Neural-fp. Neural fingerprints (Neural-fp) is constructed on a supervised deep graph convolutional neural network [34]. It applies convolutional neural networks on graphs directly. The difference between Neural-fp and Circular-fp is the replacement of the hash function. Neural-fp applies a non-linear activated densely connected layer to generate the fingerprints.

5.3.2.0.3 MPNN. Another promising graph-based deep learning techniques is the Message Passing Neural Network [73] (MPNN). It recursively updates the atom features by aggregating the feature information from its neighbors and adjacent bonds, then pools all updated features of the atoms to deliver the global representation of each molecule via a readout function. The generated representation is then fed into the downstream molecular property prediction network.

5.3.2.0.4 DMPNN. Inspired by MPNN [73], DMPNN [1] converts the passing process to bond-wise instead of atom-wise. Instead of aggregating the neighbor atoms' messages, DMPNN proposes a directed message passing scheme to avoid unnecessary loop. It aggregates the information of neighbor bonds with same direction, and takes the starter atom features as attached features to implement message passing. The following network is used to predict the property label as well.

5.3.2.0.5 $\mathsf{MV-GNN}^{\mathsf{cross}}$. $\mathsf{MV-GNN}^{\mathsf{cross}}$ model extracts the atom messages and bond messages simultaneously. It considers atom message passing and bond message passing as two parallel streams, and allows the atom/bond messages to communicate during the passing phase. A self-attention readout mechanism and a disagreement loss are employed to restrain the model training.

5.3.2.0.6 $\mathsf{MV-GNN}^{\mathsf{cross}}$ with property augmentation. The results of different models on DILI dataset empirically demonstrate $\mathsf{MV-GNN}^{\mathsf{cross}}$ has achieved the highest prediction accuracy. Considering the extremely limited availability of DILI data, we propose to involve more data in a property augmentation fashion to facilitate training the molecular representation. In this regard, we combine DILI with four more datasets with other toxicity labels to form Tox-DILI dataset, and apply $\mathsf{MV-GNN}^{\mathsf{cross}}$ model on it.

5.3.2.0.7 Additional experiments with property augmentation. In order to further proof the effectiveness of proposed method, we conduct additional experiments on Tox-DILI dataset to compare the performance improvement from using DILI only. Since MPNN and DMPNN outperform circular-fp and neural-fp on DILI dataset, and both of them are graph-based message passing models, we then utilize them to assess the prediction performance of proposed property augmentation strategy.

5.3.3 Experimental Procedure

In order to thoroughly verify the superiority of proposed method and eliminate the randomness, we have conducted extensive experiments using three evaluation methods: 5-fold cross-validation with random splitting, 10-fold leave-one-out cross-validation, and 5-fold cross-validation with scaffold splitting. To make a fair comparison, we use the same dataset splits over DILI and Tox-DILI for all the models, repectively. For each cross-validation (CV) method, we first run all the models on DILI dataset, then apply property augmentation using $\mathsf{MV-GNN}^{\mathsf{cross}}$ on the Tox-DILI dataset to further boost the performance. Moreover, we take MPNN and DMPNN as backbones to implement property augmentation to confirm the effectiveness of our method. The pair-wise comparison between experiments w/o and w/ property augmentation are visualized with a p-value calculated through the Wilcoxon test.

5.3.3.1 Cross-validation with Random Splitting

We first apply 5-fold cross-validation with random seeds to evaluate the performance of each model. In each fold, the input dataset is randomly split into 8:1:1, while 80% is used for training, 10% is used for validation, and the last 10% is used for testing. For Tox-DILI, we ensure each data split contains balanced data for each property. We calculate the mean and standard deviation of the results from all folds as the final results.

5.3.3.2 Leave-one-out Cross-validation

Considering the randomness of dataset splits in the first evaluation method, we then apply the 10-fold leave-one-out cross-validation to evaluate the performance again. The input dataset is split into 10 folds equally, each fold has been used as the testing dataset in sequence. Within the remaining 9 folds, one fold is used as the validation dataset, and the rest are used for training. We take the average of the results from all folds as the final results too.

5.3.3.3 Cross-validation with Scaffold Splitting

Other than the two commonly used evaluation methods, we also conduct experiments with scaffold splitting, which is more practical and challenging than random splitting. Scaffold splitting splits the molecules with distinct two-dimensional structural frameworks into different subsets [90], which can be considered as a clustering process based on the molecular structure prior to the training process. We follow the process introduced in [1]. The molecules in the dataset are categorized into bins based on their Murcko scaffold, which are calculated by RDKit [80]. The bins are then randomly put into train, validation and test dataset. We apply a 5 fold cross-validation here with 8:1:1 train/validation/test split too, and calculate the mean and standard deviation as the final results.

5.4 Results and Discussion

Other than the prediction accuracy, we also analysis the predicted labels with the ground truth labels in detail by computing the sensitivity, specificity, F-1 score, Matthews correlation coefficient (MCC) and ROC-AUC. All these evaluation criteria are important since we expect to find a model that can filter the drugs with potential DILI concern, as well as pick out the drugs without DILI risks, thus further experiments can be conducted on these approved drug candidates.

Table 5.5: The performance of DILI models using cross-validation with random splitting (higher is better). Best score is marked as **bold**.

	Circular-fp	Neural-fp	MPNN	DMPNN	$\mathsf{MV-GNN}^{\mathsf{cross}}$	Property Augmentation with Tox-DILI
Accuracy	$0.688_{\pm 0.051}$	$0.704_{\pm 0.091}$	$0.738_{\pm 0.094}$	$0.750_{\pm 0.098}$	$0.788_{\pm 0.077}$	$\mathbf{0.814}_{\pm 0.047}$
Sensitivity	$0.364_{\pm 0.125}$	$0.647_{\pm 0.091}$	$0.727_{\pm 0.133}$	$0.728_{\pm 0.135}$	$0.762_{\pm 0.105}$	$\mathbf{0.768}_{\pm 0.100}$
Specificity	$\mathbf{0.879}_{\pm 0.086}$	$0.740_{\pm 0.087}$	$0.752_{\pm 0.129}$	$0.764_{\pm 0.172}$	$0.809_{\pm 0.092}$	$0.849_{\pm 0.097}$
F1-score	$0.485_{\pm 0.091}$	$0.615_{\pm 0.106}$	$0.666_{\pm 0.124}$	$0.681_{\pm 0.095}$	$0.721_{\pm 0.105}$	$\mathbf{0.753}_{\pm 0.063}$
MCC	$0.289_{\pm 0.130}$	$0.381_{\pm 0.191}$	$0.473_{\pm 0.202}$	$0.499_{\pm 0.179}$	$0.562_{\pm 0.178}$	$\mathbf{0.621}_{\pm 0.114}$
ROC-AUC	$0.738_{\pm 0.056}$	$0.753_{\pm 0.093}$	$0.833_{\pm 0.075}$	$0.832_{\pm 0.068}$	$0.866_{\pm 0.055}$	$\mathbf{0.882}_{\pm 0.031}$

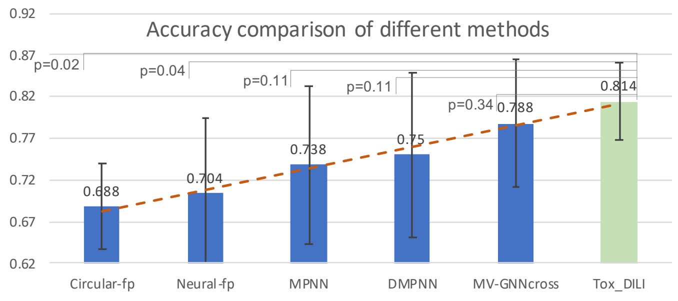


Figure 5.6: Performance comparison on accuracy of different methods using cross-validation with random splitting (higher is better). Light green color indicates our proposed method. P indicates the p-value calculated from the Wilcoxon test between our proposed method and other baselines.

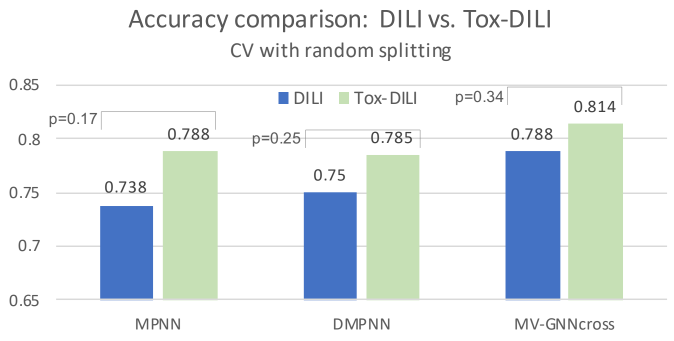


Figure 5.7: Cross-validation with random splitting. Visualization from Table 5.6. DILI indicates baseline, and Tox-DILI demonstrates the performance of utilizing property augmentation. P-value is calculated between the two prediction results for each model.

5.4.0.1 Cross-validation with Random Splitting

The prediction performance of cross-validation with random splitting are shown in Table 5.5, and visualized in Figure 5.6. As observed, graph-based message passing models generally perform better than other baselines on DILI dataset. Meanwhile, $\mathsf{MV-GNN}^{\mathsf{cross}}$ model outperforms other message passing methods, as well as equips with smaller various. The augmentation strategy that combines more data with other properties precisely improve the performance of DILI to 81.4%, which empirically proves that involving more property data to co-train the model indeed brings more information. In this fashion, $\mathsf{MV-GNN}^{\mathsf{cross}}$ model gains the accuracy boost by 2.6% compared with the vanilla $\mathsf{MV-GNN}^{\mathsf{cross}}$. The p-values obtained from the Wilcoxon test may not be sufficiently small for some baselines considering the difficulty and challenge for DILI prediction problem, yet we believe our proposed method has accomplished remarkable improvement.

As our goal is to identify drugs that might cause DILI and sort out drugs without DILI, a model with high scores of all the evaluation metric, as well as a balanced sensitivity/specificity would be more helpful. As shown in Table 5.5, Circular-fp has a very high specificity but extremely low sensitivity, so it is more likely to identify drugs without DILI as positive. The lowest MCC verifies that it cannot achieve a balanced prediction over positive and negative labels. All the criteria values of Neural-fp are not significant. MPNN and DMPNN has almost equal sensitivity and specificity scores, but the overall accuracy, F1-score and MCC are not notably high. The accuracy, sensitivity, F1-score, and MCC of $\mathsf{MV-GNN}^{\mathsf{cross}}$ are higher than other baselines on DILI dataset. The specificity score is slightly lower than Circular-fp, but is still competitive. $\mathsf{MV-GNN}^{\mathsf{cross}}$ utilizing property augmentation strategy has obtained the highest accuracy score which is 81.4%. The specificity score is fairly high as 0.849, and a sensitivity score of 0.768 is also the highest compared with other baselines. The comparisons of F1-score and MCC confirm that our $\mathsf{MV-GNN}^{\mathsf{cross}}$

Table 5.6: The performance comparison between w/o Property Augmentation (DILI) and w/ Property Augmentation (Tox-DILI) using cross-validation with random splitting. Higher score within each pair-wise comparison is marked as **bold**.

	MPNN (DILI)	MPNN (Tox-DILI)	DMPNN (DILI)	DMPNN (Tox-DILI)	$\mathsf{MV-GNN}^{\mathsf{cross}}$(DILI)	$\mathsf{MV-GNN}^{\mathsf{cross}}$(Tox-DILI)
Accuracy	$0.738_{\pm 0.094}$	$\mathbf{0.788}_{\pm 0.044}$	$0.750_{\pm 0.098}$	$\mathbf{0.785}_{\pm 0.024}$	$0.788_{\pm 0.077}$	$\mathbf{0.814}_{\pm 0.047}$
Sensitivity	$0.727_{\pm 0.133}$	$\mathbf{0.761}_{\pm 0.072}$	$0.728_{\pm 0.135}$	$\mathbf{0.748}_{\pm 0.091}$	$0.762_{\pm 0.105}$	$\mathbf{0.768}_{\pm 0.100}$
Specificity	$0.752_{\pm 0.129}$	$\mathbf{0.807}_{\pm 0.070}$	$0.764_{\pm 0.172}$	$\mathbf{0.812}_{\pm 0.045}$	$0.809_{\pm 0.092}$	$\mathbf{0.849}_{\pm 0.097}$
F1-score	$0.666_{\pm 0.124}$	$\mathbf{0.728}_{\pm 0.045}$	$\mathbf{0.764}_{\pm 0.172}$	$0.718_{\pm 0.045}$	$0.721_{\pm 0.105}$	$\mathbf{0.753}_{\pm 0.063}$
MCC	$0.473_{\pm 0.202}$	$\mathbf{0.562}_{\pm 0.082}$	$0.499_{\pm 0.179}$	$\mathbf{0.553}_{\pm 0.060}$	$0.562_{\pm 0.178}$	$\mathbf{0.621}_{\pm 0.114}$

model with property augmentation significantly perform better than other models on DILI prediction task.

We also conduct additional experiments with our method utilizing MPNN and DMPNN, where the performance is compared in Table 5.6 in a pair-wise manner (DILI vs. Tox-DILI). The accuracy improvement is visualized in Figure 5.6, and the ROC-AUC is plot in Figure 5.10. We can observe that models with proposed property augmentation almost outperform the other one over all evaluation criteria.

We can observe the performance comparison between each model based on Figure 5.10. Figure 5.10 visualizes the ROC-AUC for each model. As we know, the larger area under the curve (AUC) represents better model performance. When the inflection point is close to the left top corner, the AUC is approximate to 1. Figure **??** illustrates that $\mathsf{MV-GNN}^{\mathsf{cross}}$ on Tox-DILI outperforms other models.

Table 5.7: The performance of DILI models (higher is better) using leave-one-out cross-validation. Best score is marked as **bold**.

	Circular-fp	Neural-fp	MPNN	DMPNN	$\mathsf{MV-GNN}^{\mathsf{cross}}$	Property Augmentation with Tox-DILI
Accuracy	$0.668_{\pm 0.085}$	$0.683_{\pm 0.063}$	$0.706_{\pm 0.057}$	$0.715_{\pm 0.059}$	$0.728_{\pm 0.047}$	$\mathbf{0.787}_{\pm 0.070}$
Sensitivity	$0.351_{\pm 0.171}$	$0.595_{\pm 0.089}$	$0.590_{\pm 0.141}$	$0.617_{\pm 0.140}$	$0.651_{\pm 0.121}$	$\mathbf{0.721}_{\pm 0.106}$
Specificity	$\mathbf{0.899}_{\pm 0.063}$	$0.757_{\pm 0.081}$	$0.798_{\pm 0.115}$	$0.803_{\pm 0.107}$	$0.791_{\pm 0.087}$	$0.837_{\pm 0.062}$
F1-score	$0.447_{\pm 0.175}$	$0.604_{\pm 0.064}$	$0.614_{\pm 0.078}$	$0.631_{\pm 0.086}$	$0.655_{\pm 0.076}$	$\mathbf{0.731}_{\pm 0.076}$
MCC	$0.294_{\pm 0.120}$	$0.353_{\pm 0.118}$	$0.406_{\pm 0.114}$	$0.432_{\pm 0.113}$	$0.448_{\pm 0.099}$	$\mathbf{0.558}_{\pm 0.131}$
ROC-AUC	$0.775_{\pm 0.069}$	$0.734_{\pm 0.035}$	$0.789_{\pm 0.072}$	$0.792_{\pm 0.051}$	$0.797_{\pm 0.039}$	$\mathbf{0.840}_{\pm 0.064}$

5.4.0.2 Leave-one-out Cross-validation

To eliminate the randomness of splitting method, we use 10-fold leave-one-out cross-validation to re-run all the experiments. The performance is shown in Table 5.7 and Table 5.8. The results follow the similar trend as obtained using cross-validation with random splitting. $\mathsf{MV-GNN}^{\mathsf{cross}}$ with property augmentation learning performs best over all evaluation criteria except the specificity, where Circular-fp obtains highest value. However, the other performance results such as sensitivity, MCC and F1-score indicate that the prediction results of Circular-fp is extremely unbalanced. The accuracy and ROC-AUC visualization between w/o and w/ property augmentation on MPNN, DMPNN and $\mathsf{MV-GNN}^{\mathsf{cross}}$, which are shown in Figure 5.8 and Figure 5.11, further proof the superiority of proposed method. As shown in Figure 5.8, the p-value calculated from $\mathsf{MV-GNN}^{\mathsf{cross}}$ w/o and w/ property augmentation is less than 0.01, which can be considered as statistical significant. The prediction results with leave-one-out cross-validation confirm that our method is capable for improving the prediction performance of DILI.

5.4.0.3 Cross-validation with Scaffold Splitting

Last, we challenge the most difficult but practical scenario by conducting experiments using scaffold splitting. The results are recorded in Table 5.9 and Table 5.10, while the accuracy and ROC-AUC are visualized in Figure 5.9 and Figure 5.12. The accuracy scores have dropped compared with random splitting, which is reasonable considering the strict splitting. However, other criteria such as F1-score and MCC do not vary much, and the general trending is still similar with the performance obtained

Table 5.8: The performance comparison between w/o Property Augmentation (DILI) and w/ Property Augmentation (Tox-DILI) using leave-one-out cross-validation. Higher score within each pair-wise comparison is marked as **bold**.

	MPNN (DILI)	MPNN (Tox-DILI)	DMPNN (DILI)	DMPNN (Tox-DILI)	$\mathsf{MV-GNN}^{\mathsf{cross}}$(DILI)	$\mathsf{MV-GNN}^{\mathsf{cross}}$(Tox-DILI)
Accuracy	$0.706_{\pm 0.057}$	$\mathbf{0.736}_{\pm 0.074}$	$0.715_{\pm 0.059}$	$\mathbf{0.748}_{\pm 0.064}$	$0.728_{\pm 0.047}$	$\mathbf{0.787}_{\pm 0.070}$
Sensitivity	$0.590_{\pm 0.141}$	$\mathbf{0.625}_{\pm 0.104}$	$0.617_{\pm 0.140}$	$\mathbf{0.632}_{\pm 0.099}$	$0.651_{\pm 0.121}$	$\mathbf{0.721}_{\pm 0.106}$
Specificity	$0.798_{\pm 0.115}$	$\mathbf{0.820}_{\pm 0.117}$	$0.803_{\pm 0.107}$	$\mathbf{0.817}_{\pm 0.079}$	$0.791_{\pm 0.087}$	$\mathbf{0.837}_{\pm 0.062}$
F1-score	$0.614_{\pm 0.078}$	$\mathbf{0.655}_{\pm 0.090}$	$0.631_{\pm 0.086}$	$\mathbf{0.657}_{\pm 0.095}$	$0.655_{\pm 0.076}$	$\mathbf{0.731}_{\pm 0.076}$
MCC	$0.406_{\pm 0.114}$	$\mathbf{0.456}_{\pm 0.148}$	$0.432_{\pm 0.113}$	$\mathbf{0.454}_{\pm 0.135}$	$0.448_{\pm 0.099}$	$\mathbf{0.558}_{\pm 0.131}$
ROC-AUC	$0.789_{\pm 0.072}$	$\mathbf{0.813}_{\pm 0.070}$	$0.792_{\pm 0.051}$	$\mathbf{0.806}_{\pm 0.067}$	$0.797_{\pm 0.039}$	$\mathbf{0.840}_{\pm 0.064}$

from the other two evaluation methods. $\mathsf{MV-GNN}^{\mathsf{cross}}$ with property augmentation learning outperforms all other methods, including MPNN and DMPNN with property augmentation, which effectively illustrates the superiority of proposed method.

Table 5.9: The performance of DILI models (higher is better) using cross-validation with scaffold splitting. Best score is marked as **bold**.

	Circular-fp	Neural-fp	MPNN	DMPNN	$\mathsf{MV-GNN}^{\mathsf{cross}}$	Property Augmentation with Tox-DILI
Accuracy	$0.657_{\pm 0.037}$	$0.665_{\pm 0.048}$	$0.706_{\pm 0.010}$	$0.714_{\pm 0.043}$	$0.735_{\pm 0.045}$	$\mathbf{0.765}_{\pm 0.047}$
Sensitivity	$0.485_{\pm 0.074}$	$0.642_{\pm 0.066}$	$0.695_{\pm 0.098}$	$0.693_{\pm 0.082}$	$0.684_{\pm 0.094}$	$\mathbf{0.765}_{\pm 0.090}$
Specificity	$\mathbf{0.784}_{\pm 0.073}$	$0.688_{\pm 0.082}$	$0.708_{\pm 0.066}$	$0.724_{\pm 0.062}$	$0.765_{\pm 0.099}$	$0.774_{\pm 0.046}$
F1-score	$0.533_{\pm 0.049}$	$0.609_{\pm 0.062}$	$0.653_{\pm 0.052}$	$0.660_{\pm 0.070}$	$0.674_{\pm 0.060}$	$\mathbf{0.740}_{\pm 0.036}$
MCC	$0.284_{\pm 0.086}$	$0.328_{\pm 0.103}$	$0.402_{\pm 0.027}$	$0.415_{\pm 0.090}$	$0.458_{\pm 0.087}$	$\mathbf{0.534}_{\pm 0.089}$
ROC-AUC	$0.719_{\pm 0.028}$	$0.744_{\pm 0.051}$	$0.758_{\pm 0.025}$	$0.782_{\pm 0.040}$	$0.774_{\pm 0.042}$	$\mathbf{0.834}_{\pm 0.022}$

In addition to extensive experiments, several studies have investigated different methods to tackle DILI prediction problem in years. Recent two work, [141] and [138] also seek for appropriate approaches to enhance the prediction performance of DILIrank. [138] utilizes Bayesian model to obtain an ROC-AUC of 0.814, a sensitivity

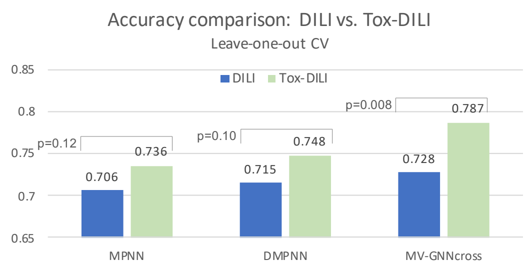


Figure 5.8: Leave-one-out cross-validation. Visualization from Table 5.8. DILI indicates baseline, and Tox-DILI demonstrates the performance of utilizing property augmentation. P-value is calculated between the two prediction results for each model.

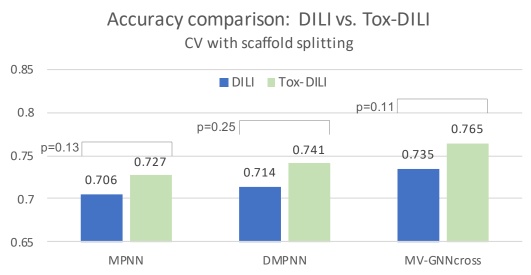


Figure 5.9: Cross-validation with scaffold splitting. Visualization from Table 5.10. DILI indicates baseline, and Tox-DILI demonstrates the performance of utilizing property augmentation. P-value is calculated between the two prediction results for each model.

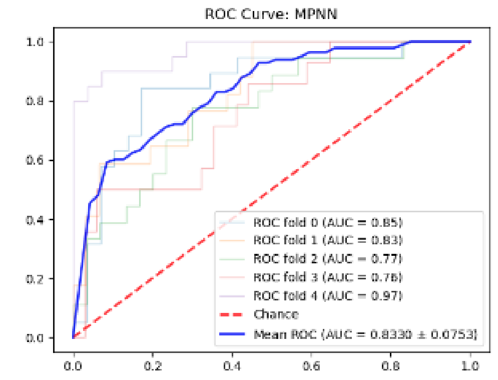
Table 5.10: The performance comparison between w/o Property Augmentation (DILI) and w/ Property Augmentation (Tox-DILI) using cross-validation with scaffold splitting. Higher score within each pair-wise comparison is marked as **bold**.

	MPNN (DILI)	MPNN (Tox-DILI)	DMPNN (DILI)	DMPNN (Tox-DILI)	$\mathsf{MV-GNN}^{\mathsf{cross}}$(DILI)	$\mathsf{MV-GNN}^{\mathsf{cross}}$(Tox-DILI)
Accuracy	$0.706_{\pm 0.010}$	**0.727** $_{\pm 0.030}$	$0.714_{\pm 0.043}$	$\mathbf{0.741}_{\pm 0.040}$	$0.735_{\pm 0.045}$	$\mathbf{0.765}_{\pm 0.047}$
Sensitivity	$0.695_{\pm 0.098}$	**0.727** $_{\pm 0.102}$	$0.693_{\pm 0.082}$	$\mathbf{0.801}_{\pm 0.073}$	$0.684_{\pm 0.094}$	$\mathbf{0.765}_{\pm 0.090}$
Specificity	$0.708_{\pm 0.066}$	**0.716** $_{\pm 0.108}$	$\mathbf{0.724}_{\pm 0.062}$	$0.669_{\pm 0.110}$	$0.765_{\pm 0.099}$	$\mathbf{0.774}_{\pm 0.046}$
F1-score	$0.653_{\pm 0.052}$	**0.717** $_{\pm 0.049}$	$0.660_{\pm 0.070}$	$\mathbf{0.748}_{\pm 0.052}$	$0.674_{\pm 0.060}$	$\mathbf{0.740}_{\pm 0.036}$
MCC	$0.402_{\pm 0.027}$	**0.452** $_{\pm 0.064}$	$0.415_{\pm 0.090}$	$\mathbf{0.482}_{\pm 0.082}$	$0.458_{\pm 0.087}$	$\mathbf{0.534}_{\pm 0.089}$
ROC-AUC	$0.758_{\pm 0.025}$	**0.796** $_{\pm 0.052}$	$0.782_{\pm 0.040}$	$\mathbf{0.814}_{\pm 0.072}$	$0.774_{\pm 0.042}$	$\mathbf{0.834}_{\pm 0.022}$

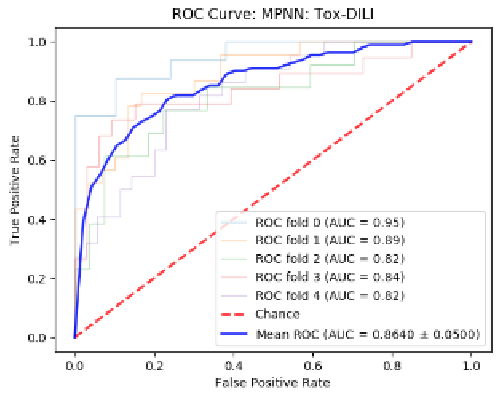
of 0.741, a specificity of 0.755, and an accuracy of 0.746. The sensitivity/specificity is nearly perfectly balanced which denotes the model holds stabilized expressive power, but the ROC-AUC and accuracy are not remarkable compared with deep graph-based models. [141] explores different features selection and various machine learning algorithms to build meta-models. Some models have achieved up to 95% sensitivity but low specificity around 50%, some models have reletively balanced sensitivity/specificity (e.g., 76%/73.2%), yet the accuracy is less than 0.75%. Ergo, it is empirically demonstrated the superior of our deep graph-based model along with property augmentation strategy.

5.5 Conclusions

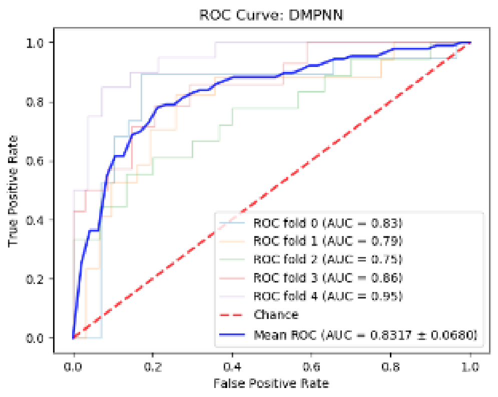
Enhancing the prediction performance of DILI is crucial for drug development. Current studies generally focus on either bringing in more features, or stacking multiple models, or enlarging the dataset. These attempts have attained impressive achievements. In spite of that, we notice that certain properties of the drugs might contain hidden correlation between each other. Hence, we propose to establish a property augmentation approach to include more information to boost the training. Extensive experiments on Tox-DILI confirm the superior of our method by improving the accuracy to 81.4% using cross-validation with random splitting, 78.7% using leave-one-out cross-validation, and 76.5% with cross-validation with scaffold splitting. Proposed method not only brings in more input data for the encoder network to learn better molecular vector representation, but also utilizes the correlations between different property labels during the prediction network. We believe it to be a promising perspective to improve the prediction performance of DILI as well as other properties with limited available data.



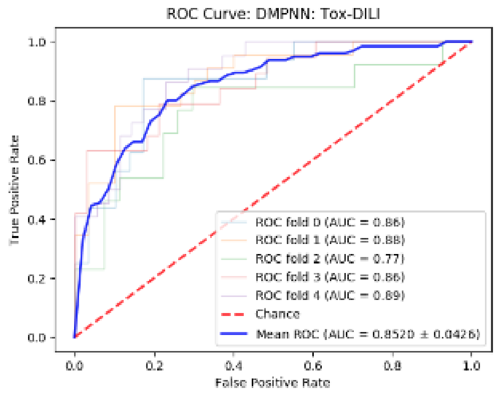
(a) MPNN on DILI.



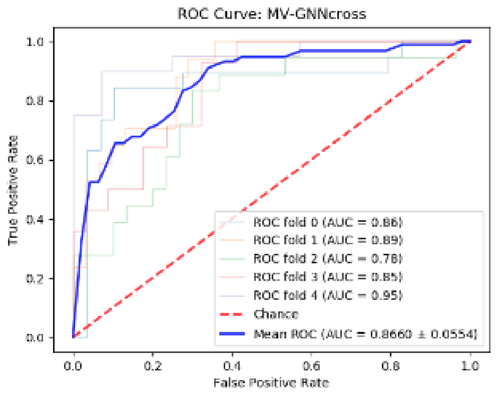
(b) MPNN on Tox-DILI.



(c) DMPNN on DILI.



(d) DMPNN on Tox-DILI.



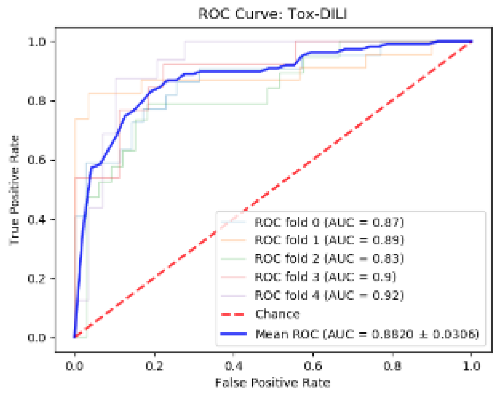
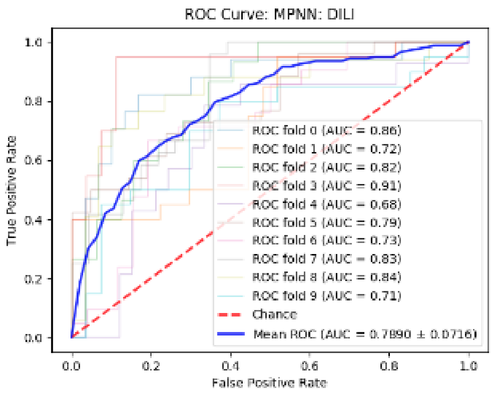
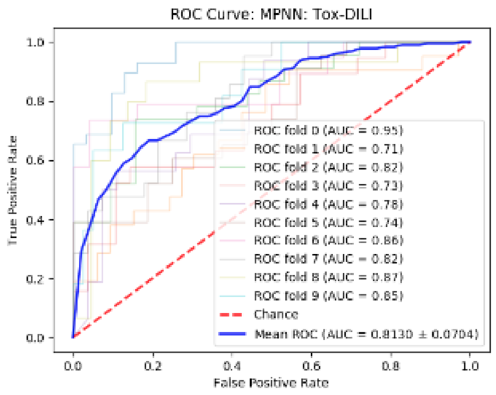


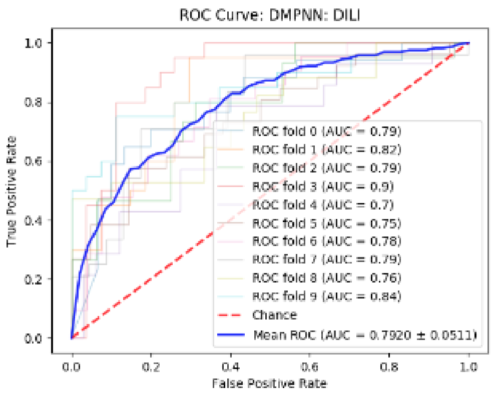
Figure 5.10: Cross-validation with random splitting. ROC Curve comparison (larger AUC is better) between w/o Property Augmentation (DILI) and w/ Property Augmentation (Tox-DILI). The lighter lines demonstrate the performance of each fold, and the blue line represents the mean AUC for each method.



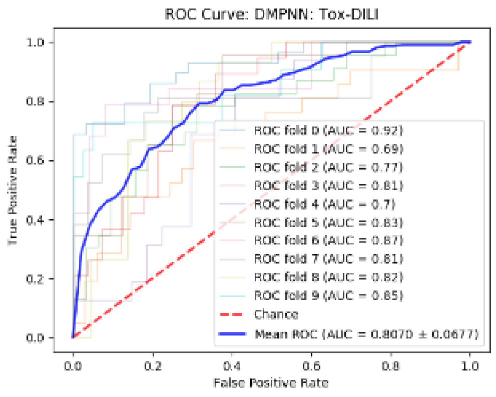
(a) MPNN on DILI.



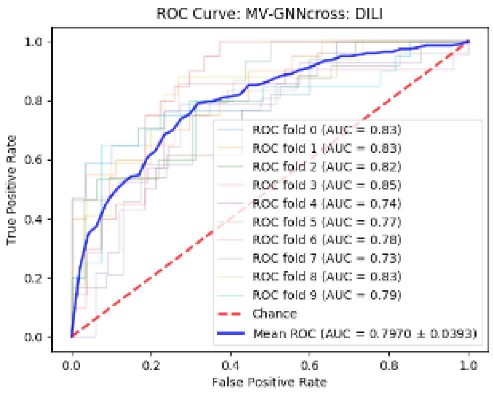
(b) MPNN on Tox-DILI.



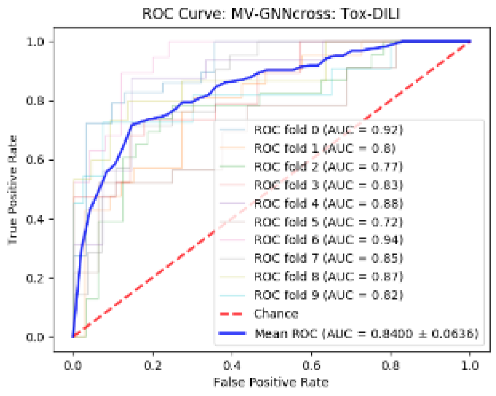
(c) DMPNN on DILI.



(d) DMPNN on Tox-DILI.

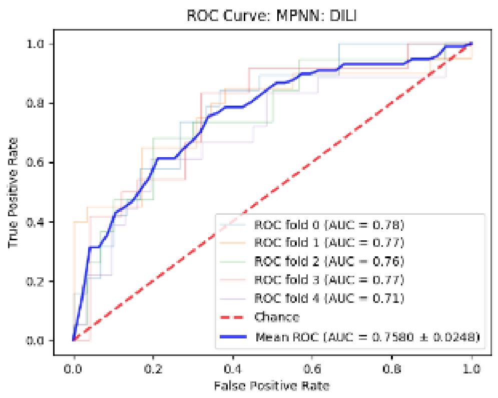


(e) MV-GNN^{cross} on DILI.

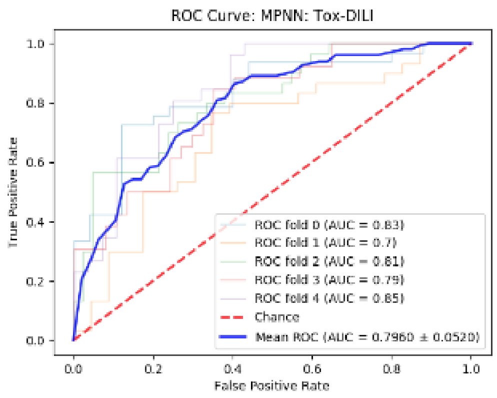


(f) MV-GNN^{cross} on Tox-DILI.

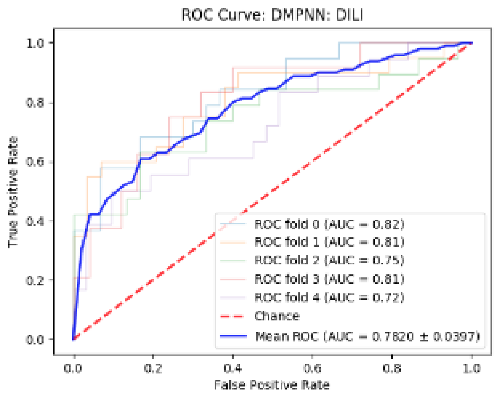
Figure 5.11: Leave-one-out cross-validation. ROC Curve comparison (larger AUC is better) between w/o Property Augmentation (DILI) and w/ Property Augmentation (Tox-DILI). The lighter lines demonstrate the performance of each fold, and the blue line represents the mean AUC for each method.



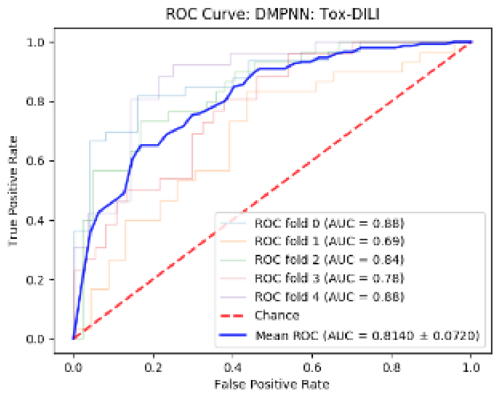
(a) MPNN on DILI.



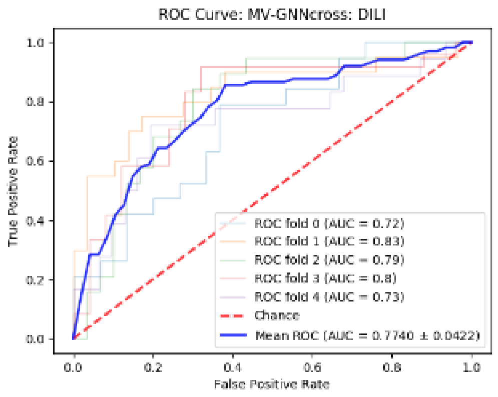
(b) MPNN on Tox-DILI.



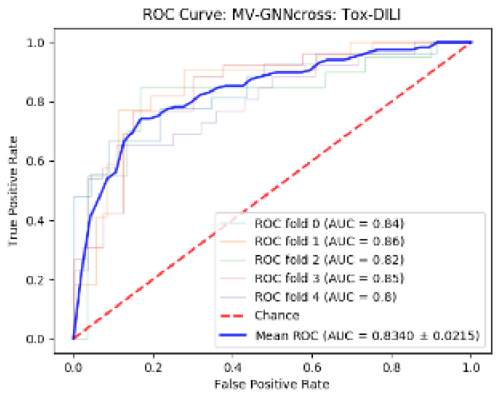
(c) DMPNN on DILI.



(d) DMPNN on Tox-DILI.



(e) MV-GNN^{cross} on DILI.



(f) MV-GNN^{cross} on Tox-DILI.

Figure 5.12: Cross-validation with scaffold splitting. ROC Curve comparison (larger AUC is better) between w/o Property Augmentation (DILI) and w/ Property Augmentation (Tox-DILI). The lighter lines demonstrate the performance of each fold, and the blue line represents the mean AUC for each method.

CHAPTER 6

IMPROVING MOLECULAR PROPERTY PREDICTION ON LIMITED DATA WITH DEEP MULTI-LABEL LEARNING

This chapter presents a multi-label learning method based on molecular SMILES sequence, which is also a method to improve the prediction performance of those properties with limited data. We propose an RNN-based multi-label molecular property prediction method to alleviate the data scarcity issue in two stages: 1) utilize the abundant unlabeled SMILES data to pre-train a seq2seq model whose encoder learns to generate molecular fingerprint based on the given SMILES; and 2) finetune the pre-trained model on the labeled molecular property data. Since labeled data is limited, we train those properties with limited sample size jointly with other properties which contain relatively sufficient samples. This approach brings in the idea of multi-label training, which is able to pre-train and fine-tune the encoder network, as well as train the prediction network with a data augmentation strategy. Extensive experiments on molecular property prediction demonstrate that our proposed method has achieved superior performance compared with the state-of-the-art approaches on properties with limited sample size.

6.1 Introduction

Molecular property prediction has been a significant task in drug discovery area. Recently, the amount of available compounds and biological activity data increased exponentially due to the experimental techniques such as High-throughput screening (HTS) and parallel synthesis [32, 33]. Effectively utilizing these large-scale chemical data would be a fruitful strategy to tackle property prediction problem. Deep learning has been widely known for its capability of taking advantage of massive amount of data [153], which leads to another surge in drug discovery domain. It would significantly save the R&D costs for drug research procedure while decreasing the failure rate in potential drug screening trials, as well as speed up the overall drug discovery process by exploiting the large-scale chemical data [45]. Lots of researches have been working on property prediction from various aspects, such as analyzing the structure of the molecule [34, 35, 109], or extracting the local features by looking into the chemical features like bounds relatives [36]. These methods generally perform prediction task by running the target property individually, and achieve good performance with sufficient labeled data. However, the prediction performance is usually unsatisfied when the labeled data is limited.

To address such problem, some previous work turns the attention to explore how to take advantage of the enormous unlabeled data. Certain attempts have successfully improved the prediction performance by adding unlabeled data as part of the training process, e.g., seq2seq and seq3seq [38, 39, 154, 155]. Nevertheless, there is another way to address this issue from a different perspective, which is to explore the potential information of limited labeled data effectively. Specifically, we propose to jointly co-train multiple properties of data, by taking those properties into the pool filled with several other properties and train them together. This multi-label idea would perform as a data augmentation for the limit-sample properties.

In this paper, we propose a data-driven Multi-label Recurrent Neural Networks based molecular property prediction technique to maximize the utilization of available data, not only unlabeled data but also the labeled data. Specifically, it can be divided into two fundamental parts, unsupervised task and supervised task. First, we applied sequence-to-sequence (seq2seq) learning on the massive unlabeled data, which is an enormous collection of molecule SMILES. It is inspired by semi-supervised learning in

Nature Language Processing (NLP) [156–158]. The SMILES sequence is input into the recurrent neural network, and converted to a vector representation called molecular fingerprint. The fingerprint then is reconstructed back to SMILES to update the network. By fully training the unlabeled data, we are able to get a pre-trained network for translating SMILES to vector with high efficiency. The intermediate fingerprint can be used for further investigation, in our case, which would be the property prediction task. The second part is to overcome the difficulty of acquiring sufficient labeled data by establishing a multi-label supervised model on a combined dataset with missing labels. As we mentioned before, training each property prediction task is ineffective and costly. In our proposed method, the input to prediction network is a data matrix with multiple property label information, which can be an original dataset collected from specialized experiments [32, 33], or formed manually by various single property. This data matrix is then fed into a prediction network to perform either classification tasks, regression task or both.

The innovation of our proposed method mainly contribute to four points: 1) It effectively utilizes the enormous unlabeled molecule data information, as well as the limit labeled molecular property data. 2) By feeding and training multiple properties data jointly into a neural network, the prediction results on the properties with limited samples are significantly improved by learning from those properties with relatively more data points. 3) The overall results on all properties perform better than training each property individually. 4) Regression and classification can be employed at the same time during our supervised training process, which assures the variety of different input labels.

6.2 Related Work

6.2.0.1 Hand-crafted Biologist-guided Hash-based Fingerprints

One conventional used traditional feature extraction method is to design the molecule fingerprint manually by experts based on biological experiments and chemical knowledge, e.g., [36, 60, 142]. This type of fingerprint methods generally work well for particular tasks but lacks universality. Hash-based methods have then been developed to address the issue of biologist-guided local-feature fingerprints. It aims to generate unique fingerprint based on different molecular features [36, 60, 159]. One critical approach is called circular fingerprint [61]. Nonetheless, it has a very notable problem, since the characteristic of the hash function is non-invertible, it might not be able to catch enough information when converting.

6.2.0.2 Sequence-based Models

SMILES sequence is a breakthrough for studying molecular property prediction by deep learning methodologies, e.g., seq2seq fingerprint [38], and seq3seq fingerprint [39]. These models spot at the potentially useful information of almost infinite molecule SMILES sequence data by adequately training them to obtain strong vector representation of the molecule. These vectors then go through other supervised models to perform property prediction, e.g., GradientBoost [65], RandomForest [160], SVM [161]. The Seq3seq fingerprint is an end-to-end semi-supervised learning method, which combines the training unlabeled data part and further prediction part together in one framework. It directly takes the generated fingerprint from the unsupervised learning network to predict the property labels.

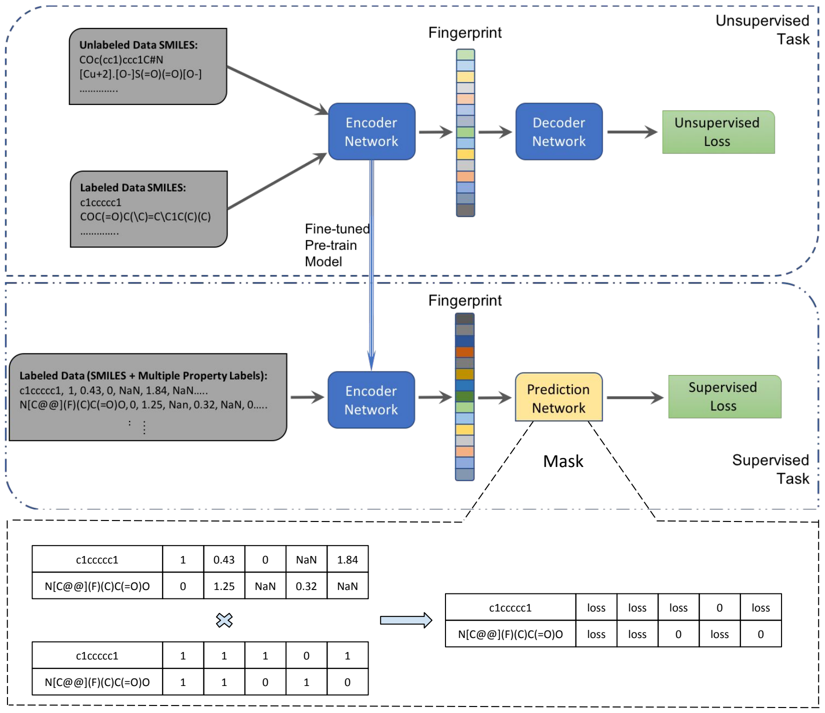

Unsupervised Task
Unlabeled Data SMILES:
COc(cc1)ccc1C#N
[Cu+2].[O-]S(=O)(=O)[O-]
Labeled Data SMILES:
c1ccccc1
COC(=O)C(\C)=C\C1C(C)(C)
Encoder Network
Fingerprint
Decoder Network
Unsupervised Loss
Fine-tuned Pre-train Model
Labeled Data (SMILES + Multiple Property Labels):
c1ccccc1, 1, 0.43, 0, NaN, 1.84, NaN.....
N[C@@](F)(C)C(=O)O, 0, 1.25, Nan, 0.32, NaN, 0.....
Encoder Network
Fingerprint
Prediction Network
Supervised Loss
Mask
Supervised Task
c1ccccc1 | 1 | 0.43 | 0 | NaN | 1.84
N[C@@](F)(C)C(=O)O | 0 | 1.25 | NaN | 0.32 | NaN
c1ccccc1 | 1 | 1 | 1 | 0 | 1
N[C@@](F)(C)C(=O)O | 1 | 1 | 0 | 1 | 0
c1ccccc1 | loss | loss | loss | 0 | loss
N[C@@](F)(C)C(=O)O | loss | loss | 0 | loss | 0

Figure 6.1: Network structure. The upper part is the unsupervised network part, and a fine-tuned encoder network from this stage is used in the lower part (the prediction network) to perform the supervised tasks.

6.3 Methodologies

Our proposed deep multi-label learning prediction contains two principal parts, unsupervised task and supervised task. The unsupervised task exploits the enormous unlabeled data, and the supervised task overcomes the dilemma of limited labeled data.

6.3.1 Network structure

The framework of our proposed method is shown in Fig. 6.1. The upper part is the unsupervised task, which is responsible for training a proper pre-trained model. SMILES data goes through the encoder network to generate a fingerprint, then enters the decoder network to recover back to SMILES sequence. The unsupervised loss is calculated and used to update the network. The encoder and decoder network share similar fundamental parts, which can be various recurrent neural network models, here in our experiment, LSTM network is implemented [162]. The pre-trained model then is employed in the following supervised prediction task, along with both SMILES and the labels as the input. A mask function is performed in the prediction network to eliminate the influence of missing labels.

6.3.2 LSTM Unit

The Long Short-Term Memory (LSTM) [162] is the most widely used recurrent neural network. LSTM has three gates: input gate, forgot gate, and output gate. A LSTM network computes a sequence of network outputs $(y'_1, \dots, y'_T)$ from the input sequence $(x_1, \dots, x_T)$ by iterating

$$f_t = \sigma_g(W_f X_t + U_f h_{t-1} + b_f) \tag{6.1}$$

$$i_t = \sigma_g(W_i x_t + U_i y'_{t-1} + b_i) \tag{6.2}$$

$$o_t = \sigma_g(W_o x_t + U_o y'_{t-1} + b_o) \tag{6.3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c + U_c h_{t-1} + b_c) \tag{6.4}$$

$$h_t = o_t \circ \sigma_h(c_t). \tag{6.5}$$

The LSTM cell has a “forgot” gate f_t which is to block some of the previous states to pass through the entire sequence. i_t and o_t are the input and output gates for the LSTM cell at time step t. c_t and h_t are the LSTM cell state and hidden state. σ represents the activation function. In our experiments, σ_g is the sigmoid function, and σ_c and σ_h are the hyperbolic tangent functions.

6.3.3 Unsupervised Pre-train Model Training

The unsupervised encoder network takes the SMILES sequence representations as the input, then outputs the corresponding vector fingerprints. The generated fingerprints then go through a decoder network to transfer the vector back to a sequence. The output sequence is compared with the input SMILES to calculate the loss, which is back-propagated to update the encoder network weights.

6.3.4 Supervised Multi-label Prediction

The pre-trained model is applied in the following supervised prediction task. Our proposed method merges multiple property datasets into a matrix with the property labels as the columns, and all observed SMILES as row index. Since different property datasets contain different number of samples, each SMILES in the matrix might only have one or some of the properties information. During the training process, when the label of the SMILES is missing, we set the associated loss as 0. By training multiple properties together, the performance is improved significantly on the properties with limited data. In addition, the prediction network is able to perform regression and classification at the same time by specifying the classification property index. The detail results are presented and discussed in the next section.

6.3.5 Loss Function

6.3.5.1 Unsupervised task

We apply the cross-entropy loss for the unsupervised task. The token vocabulary $\{v_1, v_2, \ldots, v_N\}$ of SMILES sequence is unique and limited. Set $z_t \in \mathbb{R}^N$ as the output token distribution from the LSTM cell outputs, and $l_t \in \mathbb{R}^N$ as the one-hot vector of the given original SMILES sequence token at time step t. Thus the unsupervised loss $\mathcal{L}_{unsup}$ is given by:

$$\mathcal{L}_{unsup} = \sum_{t=1}^{T} l_t^T \log(z_t). \tag{6.6}$$

6.3.5.2 Supervised task

For the supervised task, we use the **softmax** loss. It calculates the probabilities of each target class over all possible target classes. The calculated probabilities are then used to determine the target class for the given inputs.

$$\mathcal{L}_{sup} = \frac{e^{z_j}}{\sum_i e^{z_i}}. \tag{6.7}$$

The total supervised loss is the sum of each property. Property weight λ can be assigned accordingly to each property before training.

$$\mathcal{L}_{sup\_all} = \sum_{n} \mathcal{L}_{sup}. \tag{6.8}$$

6.3.6 Missing Labels Handling

In our experiment, a mask function is applied to eliminate the effects of the missing labels. After calculating the loss for each data point, the loss values are then multiplied by a matrix with the same size of the input data matrix. The mask matrix is formed by 0s and 1s, as the corresponding positions with missing labels are recorded as 0 since $w_{ij} * 0 = 0$, others are marked as 1. Thus, any weights connecting with the missing labels would have no influence on the further computation.

6.4 Experiments

6.4.1 Experiment Setup

6.4.1.1 Dataset Description

As we mentioned before, two types of datasets are used in our experiment, the unlabeled and labeled dataset are explored in the pre-training, while the labeled dataset is used in the supervised property prediction tasks. One large unlabeled dataset is used to train the encoder network, which is the ZINC Dataset [163]. It is an open-source chemical dataset which is released to the public in late 2015. ZINC contains over 35 million commercially available molecular compounds with multiple biologically relevant information, such as structure and properties. Here we pick the drug-like dataset, which contains 18,691,354 molecular in SMILES representation.

Two labeled datasets are used to perform property prediction tasks, NIH-17p and NCI. Within them, NIH-17p is used for both classification tasks and regression tasks, while NCI is only used in the regression tasks based on the label characteristic.

- NIH-17p is provided by the National Center for Advancing Translational Sciences (NCATS) at National Institutes of Health (NIH). It consists of 17 individual properties associated with molecular SMILES representations. The sample size of each property is different (shown under the "Sample-Numbers" in Table 6.1 and Table 6.2); some of them have very limited data. For instance, property **pgb** has only **186** samples, **heptox** has **440** objects, and **vd** has **668** data points. On the other hand, some properties have up to almost 100 times data compared with them, e.g., the sample size of 2d6 is 15428, and 1a2 is 14226. NIH-17p contains both continuous labels (3/17) and binary labels (14/17). The classification tasks are established on the 14 properties, and the regression tasks are conducted on the remaining three properties. We randomly split the dataset into training, validation, and testing by the ratio of 80%, 10%, and 10%.

- NCI dataset is download from Deepchem [164]. It contains 11,738 unique molecule SMILES with eight cancer-related property labels, which are CCRE, HL-60, K-526, RPMI, A549, COLO, HCC, MALME. Since no missing labels are observed in NCI dataset, we manually generate a new dataset from the original NCI dataset with missing labels. The detailed approach is selecting two properties, says CCRE and HL-60, and randomly set 95% and 90% of the labels as missing, other properties remain unchanged. Next, the multi-label prediction is applied on this dataset to evaluate our proposed assumption, which is properties with more data would help improve the performance of the properties with limited data. The dataset is split by 80% training, 10% validating, and 10% testing as well. We repeat this approach 4 times (each time sets two different properties as the properties with "limited data") to eliminate the contingency.

Table 6.1: The regression results of NIH-17p (RMSE, lower is better).

Property	Sample-Numbers	Circular-FP	Neural-FP	Seq2seq-FP	Seq3seq-FP	**Ours**
Logp	10851	1.3437	0.7085	1.4065	0.5864	**0.4534**
pampar	4071	0.7609	0.6517	0.7343	0.6959	**0.5578**
vd	668	0.5870	0.5789	0.6321	0.6183	**0.5015**

6.4.2 Hyper-parameters Settings

For the unsupervised task, the encoder network is a 3 layers LSTM network with the input embedding dimension as 128, and the hidden size as 256. The decoder network is assigned with the same hyper-parameters with the encoder network, with

Table 6.2: The classification results on NIH-17p data (accuracy, higher is better).

Property	Sample-Numbers	Circular-FP	Neural-FP	Graph-Representation	Seq2seq-FP	Seq3seq-FP	**Ours**
pgb	186	**72.22%**	66.67%	61.11%	**72.22%**	66.67%	**72.22%**
solub	6694	82.33%	80.17%	81.38%	62.64%	77.87%	**83.05%**
2c9	13064	78.05%	80.70%	**81.65%**	67.27%	79.38%	80.55%
2d6	15428	87.88%	88.08%	77.58%	86.38%	88.79%	**89.25%**
2c19	11833	76.15%	80.43%	**83.02%**	66.24%	77.44%	81.45%
rlm	10626	78.82%	73.03%	77.91%	62.58%	75.14%	**80.17%**
mmp	5970	88.40%	89.22%	86.79%	86.93%	87.91%	**90.36%**
ames	8224	**79.95%**	78.91%	78.59%	63.39%	74.97%	79.18%
pldc	4161	90.50%	**93.75%**	81.21%	91.75%	93.00%	92.75%
heptox	440	75.93%	62.96%	79.63%	62.96%	62.96%	**81.48%**
3a4	13433	78.57%	78.50%	78.29%	69.60%	75.30%	**79.64%**
pampac	4698	77.28%	76.88%	77.87%	68.56%	73.23%	**83.98%**
herg	3024	90.81%	91.17%	85.96%	85.16%	90.11%	**91.52%**
1a2	14226	83.38%	82.61%	83.88%	70.97%	84.78%	**85.06%**

the output dimension as 128. The optimizer used is Adam, and the dropout rate is 0.5. The supervised model is one layer fully connected neural network.

6.4.3 Evaluation Metric

Exact match accuracy and the cross-entropy loss are used to back-propagate and update the unsupervised neural network. The exact match accuracy is a measure of the portion of accurately recovered sequence within the entire samples. For the followed supervised tasks, the root mean squared error (RMSE) is used to measure the performance of the regression task, and the accuracy is used for validating the classification results.

6.4.4 Comparison Experiments

6.4.4.1 Regression task: NIH-17p.

The comparison experiments on regression task is conducted on both NIH-17p dataset and NCI dataset with four baseline methods, circular fingerprint (circular-FP) [61], neural fingerprint (neural-FP) [34], seq2seq fingerprint (seq2seq) [38], and seq3seq fingerprint (seq3seq) [39] . The circular-FP is generated by a hand-crafted hash-based algorithm to define the local features. The neural-FP is constructed by a supervised deep graph convolutional neural network. Seq2seq fingerprint and seq3seq fingerprint are RNN based models.

6.4.4.2 Regression task: NCI with missings.

Regression tasks are running on NCI dataset with missings too. We picked two properties in the proper order to conduct four experiments, each contains one property with 95% manually assigned missings, and one property with 90% missings, others remain still. The details about the data sample numbers can be found in Table 6.3 - Table 6.6. Three comparison methods are applied on these datasets: circular-FP, neural-FP, and seq3seq fingerprint. Due to the limited position in the paper, we take out seq2seq since seq3seq fingerprint generally performs better.

6.4.4.3 Classification task.

One recently published method, molecular properties prediction utilizing graph-level representation (Graph-Representation) [165], has been added to perform classification task. This method proposed an idea of presenting molecular properties by learning graph-level features instead of node-level. Since it can only predict positive and negative labels, it is not included in the regression tasks. The dataset used for the classification experiments is formed by the 14 properties from NIH-17p dataset, which carry binary labels. Logp, pampar, and vd are excluded in the classification tasks, and used to conduct the regression tasks.

6.4.5 Experiment Results

The results of all the comparison methods and our proposed model are shown in the following tables. Table 6.1 is the RMSE results of running different models on

Table 6.3: The RMSE results of the regression tasks on the NCI dataset: CCRE & HL-60. Properties with limited data are marked as grey (lower is better).

Prop (sample#)	Circular	Neural	Seq2seq	Seq3seq	Ours
CCRE (576)	0.8104	0.8591	0.8108	0.8218	**0.7914**
HL-60 (1134)	0.7464	0.6538	0.6704	0.6569	**0.6335**
K-526 (11738)	**1.0530**	1.0778	1.0694	1.0685	1.0587
RPMI (11738)	1.1251	1.0231	1.0371	1.0353	**1.0203**
A549 (11738)	**0.7543**	0.7571	0.7673	0.7563	0.7549
COLO (11738)	0.7554	0.7787	**0.7546**	0.7634	0.7605
HCC (11738)	0.7074	0.7115	0.7125	0.7048	**0.7035**
MALME (11738)	0.7673	0.7653	0.7683	0.7710	**0.7582**

Table 6.4: The RMSE results of the regression tasks on the NCI dataset: K-256 & RPMI. Properties with limited data are marked as grey (lower is better).

Prop (sample#)	Circular	Neural	Seq2seq	Seq3seq	Ours
CCRE (11738)	0.9838	0.9896	0.9975	0.9921	**0.9828**
HL-60 (11738)	0.8068	0.8085	0.8077	0.8099	**0.8051**
K-526 (579)	0.9059	1.2606	0.9160	0.8759	**0.8548**
RPMI (1180)	1.1251	1.1329	1.0814	1.0778	**1.0654**
A549 (11738)	**0.7543**	0.7571	0.7673	0.7563	0.7549
COLO (11738)	0.7554	0.7787	**0.7546**	0.7634	0.7605
HCC (11738)	0.7074	0.7115	0.7125	0.7048	**0.7035**
MALME (11738)	0.7673	0.7653	0.7683	0.7710	**0.7582**

Table 6.5: The RMSE results of the regression tasks on the NCI dataset: A549 & COLO. Properties with limited data are marked as grey (lower is better).

Prop (sample#)	Circular	Neural	Seq2seq	Seq3seq	Ours
CCRE (11738)	**0.9838**	0.9872	0.9975	0.9961	**0.9838**
HL-60 (11738)	0.8063	0.8085	0.8077	0.8123	**0.8043**
K-526 (11738)	**1.0530**	1.0860	1.0694	1.0689	1.0605
RPMI (11738)	1.1251	1.0276	1.0371	1.0292	**1.0217**
A549 (617)	0.8559	0.8840	0.8974	0.8531	**0.8475**
COLO (1191)	0.8244	0.8243	0.8130	0.8109	**0.8056**
HCC (11738)	0.7074	0.7115	0.7125	0.7083	**0.7024**
MALME (11738)	0.7673	0.7680	0.7683	0.7719	**0.7600**

NIH-17p dataset, Table 6.2 is the accuracy results of the 14 properties from NIH-17p dataset. Table 6.3 - 6.6 shows the results of the extensive experiments on NCI dataset with randomly assigning missings, the evaluation criterion is RMSE since all labels in the NCI dataset are continuous.

Table 6.6: The RMSE results of the regression tasks on the NCI dataset: HCC & MALME. Properties with limited data are marked as grey (lower is better).

Prop (sample#)	Circular	Neural	Seq2seq	Seq3seq	Ours
CCRE (11738)	0.9838	0.9896	0.9975	0.9903	**0.9813**
HL-60 (11738)	0.8063	**0.8025**	0.8077	0.8104	0.8073
K-526 (11738)	**1.0530**	1.0778	1.0694	1.0682	1.0602
RPMI (11738)	1.1251	1.0231	1.0371	1.0299	**1.0218**
A549 (11738)	0.7543	0.7571	0.7673	0.7636	**0.7524**
COLO (11738)	0.7554	0.7787	**0.7546**	0.7674	0.7611
HCC (603)	0.7769	0.7364	0.6841	0.6813	**0.6604**
MALME (1128)	0.9930	1.0233	0.9861	0.9940	**0.9707**

As observed, Table 6.1 and table 6.2 clearly show that properties with limited data have achieved up to 13.4% improvement, which are **vd** (668 samples), **heptox** (440 samples), and **pgb** (186 samples). For other properties, the overall results are also better than the baseline methods. Those properties with more data samples

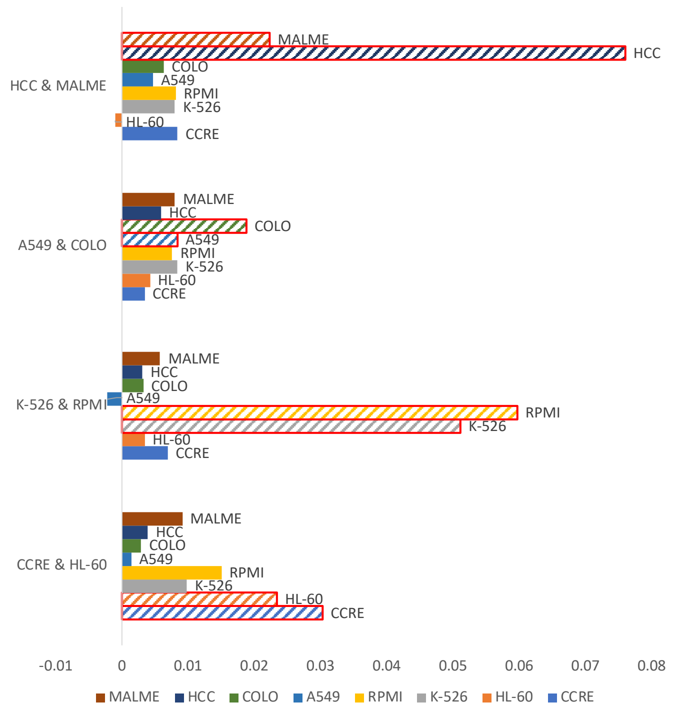


Figure 6.2: Results Comparison of the NCI dataset. The figure shows the RMSE difference between our proposed method and the best baseline method in the four NCI experiments.

rarely improve since overall deep learning methods are able to obtain good results by training sufficient data adequately. We argue that the performance of pgb is same as the other two baselines probably due to the extremely small sample size (only 186). The sample size of the other two properties appear to be appropriate for multi-label training (approximately hundreds of samples).

The comparison experiments on the revised NCI dataset fully prove the effectiveness of our proposed method. By randomly assigning large proportion of missing labels to certain properties, which manually forms a dataset contains properties with limited data and properties with sufficient data, has confirmed that proposed multi-label learning is able to extract more information than training each property individually, especially boost the performance of the properties with limited data. The four runs with different "limited data" properties demonstrate the robustness of our method. Overall, training multiple properties together indeed improve the overall performance compared with training individual property.

6.5 Conclusion

In this paper, we propose an innovative idea of combining and training multiple labeled datasets together to perform deep multi-label learning for molecular property prediction. Our method takes advantages of both labeled data and unlabeled data in a multi-label learning fashion, which enables effectively improve the prediction performance from two perspectives, 1) we draw abundant information from vast unlabeled SMILES data to obtain a good encoder model for generating accurate fingerprints, and 2) we exploit labeled data by coordinately training multiple properties to promote the performance of small-sized properties. The results of extensive experiments demonstrate the effectiveness and robustness of our method, which can significantly improve the performance of properties with limited data.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Molecular property prediction problem has been studied for many years. The problem itself is not complicated, given a molecule, researchers conduct experiments or computers run models to test whether it contains certain properties. In terms of solving it with deep learning techniques, the progress can be summarized as data preparation, molecular representation learning, and property prediction. During my Ph.D. studies, I made efforts to research each step and discovered several perspectives that can be improved to promote prediction performance: 1) design powerful molecular representation learning models; 2) consider the effects of data distribution; and 3) explore unlabeled data to assist the supervised prediction tasks. My studies have demonstrated that deep learning techniques are a proper way to tackle the molecular property prediction problem. In specific, the following methods have been developed, and the extensive experiment results have validated the effectiveness.

Cross-dependent graph neural networks for molecular property prediction presents a novel cross-dependent graph neural network with the multi-view architecture. Unlike previous methods, our approach integrates atom and bond information through a multi-view perspective and incorporates a cross-dependent message passing mechanism, ensuring updated information flow and enhancing both efficiency and expressive power.

Gradient-norm based attentive loss for molecular property prediction propose to utilize a self-attention mechanism to generate a learnable weight for each data sample according to the associated gradient norm. The learned attention value is then embedded into the prediction models to construct an attentive loss for network updating and back-propagation.

Robust self-training strategy for various molecular biology prediction tasks utilizes self-training paradigm to explore the large amount unlabeled molecules information, and further equips with robust loss to handle the label-noise problem during the training.

Deep graph learning with property augmentation for predicting drug-induced liver injury tackles a practical problem in drug discovery, which is DILI prediction. With the help of our proposed property augmentation strategy with toxicity property data, additional information is involved to enhance network training. This approach not only enhances the learning of molecular representation through increased input data but also exploits correlations between property labels, holding potential to enhance not only DILI prediction but also other properties with limited data.

Improving molecular property prediction on limited data with deep multi-label learning is implemented with sequence-based models along with multi-label learning, which further demonstrates the effectiveness of multi-label learning in enhancing prediction performance from two angles: firstly, by extracting valuable information from extensive unlabeled SMILES data to establish an accurate encoder model for representation learning, and secondly, by jointly training multiple properties using labeled data to enhance the performance of smaller properties. Extensive experimental results substantiate the effectiveness of our approach, showcasing significant improvements for properties with limited available data.

For my next step, I will extend my knowledge and keep working in this area. With more types of information available, i.e., molecular image information and 3D structure, how to effectively utilize and integrate various types of data is a potential direction to develop more advanced molecular representation learning models.

Moreover, with the rapid development of large language models (LLMs), it is an emerging field that has shown remarkable capabilities in understanding complex biological and chemical contexts. However, how to adapt it to molecular property prediction is still a challenge. I am enthusiastic about exploring these cutting-edge techniques and integrating them into my research topic.

REFERENCES

[1] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, *et al.*, "Analyzing learned molecular representations for property prediction," *Journal of chemical information and modeling*, vol. 59, no. 8, pp. 3370–3388, 2019.

[2] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific data*, vol. 1, no. 1, pp. 1–7, 2014.

[3] S. M. Paul, D. S. Mytelka, C. T. Dunwiddie, C. C. Persinger, B. H. Munos, S. R. Lindborg, and A. L. Schacht, "How to improve r&d productivity: the pharmaceutical industry's grand challenge," *Nature reviews Drug discovery*, vol. 9, no. 3, pp. 203–214, 2010.

[4] M. Dickson and J. P. Gagnon, "Key factors in the rising cost of new drug discovery and development," *Nature reviews Drug discovery*, vol. 3, no. 5, p. 417, 2004.

[5] X. Chen, C. C. Yan, X. Zhang, X. Zhang, F. Dai, J. Yin, and Y. Zhang, "Drug–target interaction prediction: databases, web servers and computational models," *Briefings in bioinformatics*, vol. 17, no. 4, pp. 696–712, 2015.

[6] R. Chen, X. Liu, S. Jin, J. Lin, and J. Liu, "Machine learning for drug-target interaction prediction," *Molecules*, vol. 23, no. 9, p. 2208, 2018.

[7] J. P. Hughes, S. Rees, S. B. Kalindjian, and K. L. Philpott, "Principles of early drug discovery," *British journal of pharmacology*, vol. 162, no. 6, pp. 1239–1249, 2011.

[8] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] J. Yang, J. Duan, S. Tran, Y. Xu, S. Chanda, L. Chen, B. Zeng, T. Chilimbi, and J. Huang, “Vision-language pre-training with triple contrastive learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 671–15 680.

[11] J. Yang, C. Li, W. An, H. Ma, Y. Guo, Y. Rong, P. Zhao, and J. Huang, “Exploring robustness of unsupervised domain adaptation in semantic segmentation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9194–9203.

[12] J. Yang, W. An, S. Wang, X. Zhu, C. Yan, and J. Huang, “Label-driven reconstruction for domain adaptation in semantic segmentation,” in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16.* Springer, 2020, pp. 480–498.

[13] J. Yang, P. Zhao, Y. Rong, C. Yan, C. Li, H. Ma, and J. Huang, “Hierarchical graph capsule network,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 603–10 611.

[14] M. M. Haq, H. Ma, and J. Huang, “Nusegda: Domain adaptation for nuclei segmentation,” *Frontiers in Big Data*, vol. 6, p. 1108659, 2023.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.

[16] K. Ding, Q. Liu, E. Lee, M. Zhou, A. Lu, and S. Zhang, “Feature-enhanced graph networks for genetic mutational prediction using histopathological images in

colon cancer," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II 23*. Springer, 2020, pp. 294–304.

[17] K. Ding, M. Zhou, Z. Wang, Q. Liu, C. W. Arnold, S. Zhang, and D. N. Metaxas, "Graph convolutional networks for multi-modality medical imaging: Methods, architectures, and clinical applications," *arXiv preprint arXiv:2202.08916*, 2022.

[18] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "Joint learning of words and meaning representations for open-text semantic parsing," in *Artificial Intelligence and Statistics*, 2012, pp. 127–135.

[19] R. Socher, "Recursive deep learning for natural language processing and computer vision," Ph.D. dissertation, Citeseer, 2014.

[20] K. Ding, M. Zhou, H. Wang, S. Zhang, and D. N. Metaxas, "Spatially aware graph neural networks and cross-level molecular profile prediction in colon cancer histopathology: a retrospective multi-cohort study," *The Lancet Digital Health*, vol. 4, no. 11, pp. e787–e795, 2022.

[21] K. Ding, M. Zhou, H. Wang, O. Gevaert, D. Metaxas, and S. Zhang, "A large-scale synthetic pathological dataset for deep learning-enabled segmentation of breast cancer," *Scientific Data*, vol. 10, no. 1, p. 231, 2023.

[22] K. Ding, M. Zhou, D. N. Metaxas, and S. Zhang, "Pathology-and-genomics multimodal transformer for survival outcome prediction," *arXiv preprint arXiv:2307.11952*, 2023.

[23] W. An, Y. Guo, Y. Bian, H. Ma, J. Yang, C. Li, and J. Huang, "Modna: motif-oriented pre-training for dna language model," in *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 2022, pp. 1–5.

[24] Y. Guo, J. Wu, H. Ma, S. Wang, and J. Huang, "Bagging msa learning: Enhancing low-quality pssm with deep learning for accurate protein structure property prediction," in *Research in Computational Molecular Biology: 24th Annual International Conference, RECOMB 2020, Padua, Italy, May 10–13, 2020, Proceedings 24.* Springer, 2020, pp. 88–103.

[25] ——, "Eptool: a new enhancing pssm tool for protein secondary structure prediction," *Journal of Computational Biology*, vol. 28, no. 4, pp. 362–364, 2021.

[26] Y. Guo, J. Wu, H. Ma, and J. Huang, "Self-supervised pre-training for protein embeddings using tertiary structures," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, 2022, pp. 6801–6809.

[27] Y. Guo, J. Wu, H. Ma, S. Wang, and J. Huang, "Protein ensemble learning with atrous spatial pyramid networks for secondary structure prediction," in *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).* IEEE, 2020, pp. 17–22.

[28] ——, "Comprehensive study on enhancing low-quality position-specific scoring matrix with deep learning for accurate protein structure property prediction: using bagging multiple sequence alignment learning," *Journal of Computational Biology*, vol. 28, no. 4, pp. 346–361, 2021.

[29] Y. Guo, J. Wu, H. Ma, J. Yang, X. Zhu, and J. Huang, "Weightaln: Weighted homologous alignment for protein structure property prediction," in *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).* IEEE, 2020, pp. 72–75.

[30] Y. Guo, J. Wu, H. Ma, S. Wang, and J. Huang, "Deep ensemble learning with atrous spatial pyramid networks for protein secondary structure prediction," *Biomolecules*, vol. 12, no. 6, p. 774, 2022.

[31] C. Yan, J. Yang, H. Ma, S. Wang, and J. Huang, "Molecule sequence generation with rebalanced variational autoencoder loss," *Journal of Computational Biology*, vol. 30, no. 1, pp. 82–94, 2023.

[32] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, "The rise of deep learning in drug discovery," *Drug discovery today*, vol. 23, no. 6, pp. 1241–1250, 2018.

[33] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, *et al.*, "Pubchem substance and compound databases," *Nucleic acids research*, vol. 44, no. D1, pp. D1202–D1213, 2015.

[34] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.

[35] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," *arXiv preprint arXiv:1801.03226*, 2018.

[36] R. C. Glen, A. Bender, C. H. Arnby, L. Carlsson, S. Boyer, and J. Smith, "Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to adme," *IDrugs*, vol. 9, no. 3, p. 199, 2006.

[37] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[38] Z. Xu, S. Wang, F. Zhu, and J. Huang, "Seq2seq fingerprint: An unsupervised deep molecular embedding for drug discovery," in *BCB*, 2017.

[39] X. Zhang, S. Wang, F. Zhu, Z. Xu, Y. Wang, and J. Huang, "Seq3seq fingerprint: towards end-to-end semi-supervised deep drug discovery," in *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 2018, pp. 404–413.

[40] H. Ma, Y. Bian, Y. Rong, W. Huang, T. Xu, W. Xie, G. Ye, and J. Huang, "Cross-dependent graph neural networks for molecular property prediction," *Bioinformatics*, vol. 38, no. 7, pp. 2003–2009, 2022.

[41] H. Ma, Y. Rong, B. Liu, Y. Guo, C. Yan, and J. Huang, "Gradient-norm based attentive loss for molecular property prediction," in *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).* IEEE, 2021, pp. 497–502.

[42] H. Ma, F. Jiang, Y. Rong, Y. Guo, and J. Huang, "Robust self-training strategy for various molecular biology prediction tasks," in *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 2022, pp. 1–5.

[43] H. Ma, W. An, Y. Wang, H. Sun, R. Huang, and J. Huang, "Deep graph learning with property augmentation for predicting drug-induced liver injury," *Chemical Research in Toxicology*, vol. 34, no. 2, pp. 495–506, 2020.

[44] H. Ma, C. Yan, Y. Guo, S. Wang, Y. Wang, H. Sun, and J. Huang, "Improving molecular property prediction on limited data with deep multi-label learning," in *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM).* IEEE, 2020, pp. 2779–2784.

[45] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, *et al.*, "Opportunities and obstacles for deep learning in biology and medicine," *Journal of The Royal Society Interface*, vol. 15, no. 141, p. 20170387, 2018.

[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[47] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *NeurIPS*, 2018, pp. 4558–4567.

[48] K. Guu, J. Miller, and P. Liang, "Traversing knowledge graphs in vector space," *arXiv preprint arXiv:1506.01094*, 2015.

[49] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec, "Embedding logical queries on knowledge graphs," in *NeurIPS*, 2018, pp. 2026–2037.

[50] M. Mao, J. Lu, G. Zhang, and J. Zhang, "Multirelational social recommendations via multigraph ranking," *IEEE transactions on cybernetics*, vol. 47, no. 12, pp. 4049–4061, 2016.

[51] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *NeurIPS*, 2017, pp. 3697–3707.

[52] G. Neglur, R. L. Grossman, and B. Liu, "Assigning unique keys to chemical compounds for data integration: Some interesting counter examples," in *International Workshop on Data Integration in the Life Sciences.* Springer, 2005, pp. 145–157.

[53] D. E. Pires, T. L. Blundell, and D. B. Ascher, "pkcsm: predicting small-molecule pharmacokinetic and toxicity properties using graph-based signatures," *Journal of medicinal chemistry*, vol. 58, no. 9, pp. 4066–4072, 2015.

[54] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," in *NeurIPS*, 2017, pp. 6530–6539.

[55] S. K. Bhal, "Logp—making sense of the value," *Advanced Chemistry Development, Toronto, ON, Canada*, pp. 1–4, 2007.

[56] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[57] B. Weisfeiler and A. A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

[58] K. Preuer, G. Klambauer, F. Rippmann, S. Hochreiter, and T. Unterthiner, "Interpretable deep learning in drug discovery," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning.* Springer, 2019, pp. 331–345.

[59] J. Zhao, X. Xie, X. Xu, and S. Sun, "Multi-view learning overview: Recent progress and new challenges," *Information Fusion*, vol. 38, pp. 43–54, 2017.

[60] H. Morgan, "The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service," *J. Chemical Documentation*, vol. 5, pp. 107–113, 1965.

[61] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of chemical information and modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[62] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression.* Springer, 2002.

[63] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[64] S. J. Swamidass, C.-A. Azencott, T.-W. Lin, H. Gramajo, S.-C. Tsai, and P. Baldi, "Influence relevance voting: an accurate and interpretable virtual high throughput screening method," *Journal of chemical information and modeling*, vol. 49, no. 4, pp. 756–766, 2009.

[65] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[66] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: a benchmark for molecular machine learning," *Chemical Science*, vol. 9, no. 2, pp. 513–530, 2018.

[67] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of computer-aided molecular design*, vol. 30, no. 8, pp. 595–608, 2016.

[68] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature communications*, vol. 8, p. 13890, 2017.

[69] K. Schütt, P.-J. Kindermans, H. E. S. Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," in *NeurIPS*, 2017, pp. 991–1001.

[70] S. Ryu, J. Lim, S. H. Hong, and W. Y. Kim, "Deeply learning molecular structure-property relationships using attention-and gate-augmented graph convolutional network," *arXiv preprint arXiv:1805.10988*, 2018.

[71] Z. Xiong, D. Wang, X. Liu, F. Zhong, X. Wan, X. Li, Z. Li, X. Luo, K. Chen, H. Jiang, *et al.*, "Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism," *Journal of Medicinal Chemistry*, 2019.

[72] X. Jiang, P. Ji, and S. Li, "Censnet: Convolution with edge-node switching in graph neural networks." in *IJCAI*, 2019, pp. 2656–2662.

[73] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*. JMLR. org, 2017, pp. 1263–1272.

[74] J. Klicpera, J. Groß, and S. Günnemann, "Directional message passing for molecular graphs," in *ICLR*, 2020.

[75] Y. Song, S. Zheng, Z. Niu, Z.-H. Fu, Y. Lu, and Y. Yang, "Communicative representation learning on attributed molecular graphs," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence,(IJCAI 2020)*, 2020, pp. 2831–2838.

[76] C. Lu, Q. Liu, C. Wang, Z. Huang, P. Lin, and L. He, “Molecular property prediction: A multilevel quantum interactions modeling perspective,” in *Proceedings of AAAI*, vol. 33, 2019, pp. 1052–1060.

[77] A. M. Richard, R. S. Judson, K. A. Houck, C. M. Grulke, P. Volarath, I. Thillainadarajah, C. Yang, J. Rathman, M. T. Martin, J. F. Wambaugh, *et al.*, “Toxcast chemical landscape: paving the road to 21st century toxicology,” *Chemical research in toxicology*, vol. 29, no. 8, pp. 1225–1251, 2016.

[78] I. F. Martins, A. L. Teixeira, L. Pinheiro, and A. O. Falcao, “A bayesian approach to in silico blood-brain barrier penetration modeling,” *Journal of chemical information and modeling*, vol. 52, no. 6, pp. 1686–1697, 2012.

[79] R. Ramakrishnan, M. Hartmann, E. Tapavicza, and O. A. Von Lilienfeld, “Electronic spectra from tddft and machine learning in chemical space,” *The Journal of chemical physics*, vol. 143, no. 8, p. 084111, 2015.

[80] G. Landrum *et al.*, “Rdkit: Open-source cheminformatics,” 2006.

[81] F. Harary and R. Z. Norman, “Some properties of line digraphs,” *Rendiconti del Circolo Matematico di Palermo*, vol. 9, no. 2, pp. 161–168, May 1960. [Online]. Available: https://doi.org/10.1007/BF02854581

[82] J. Li, Y. Rong, H. Cheng, H. Meng, W. Huang, and J. Huang, “Semi-supervised graph classification: A hierarchical graph perspective,” in *TheWebConf*. ACM, 2019, pp. 972–982.

[83] G. Subramanian, B. Ramsundar, V. Pande, and R. A. Denny, “Computational modeling of β-secretase 1 (bace-1) inhibitors using ligand based approaches,” *Journal of chemical information and modeling*, vol. 56, no. 10, pp. 1936–1949, 2016.

[84] M. Kuhn, I. Letunic, L. J. Jensen, and P. Bork, “The sider database of drugs and side effects,” *Nucleic acids research*, vol. 44, no. D1, pp. D1075–D1079, 2015.

[85] K. M. Gayvert, N. S. Madhukar, and O. Elemento, “A data-driven approach to predicting successes and failures of clinical trials,” *Cell chemical biology*, vol. 23, no. 10, pp. 1294–1301, 2016.

[86] L. C. Blum and J.-L. Reymond, “970 million druglike small molecules for virtual screening in the chemical universe database GDB-13,” *J. Am. Chem. Soc.*, vol. 131, p. 8732, 2009.

[87] J. S. Delaney, “Esol: estimating aqueous solubility directly from molecular structure,” *Journal of chemical information and computer sciences*, vol. 44, no. 3, pp. 1000–1005, 2004.

[88] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, *et al.*, “Chembl: a large-scale bioactivity database for drug discovery,” *Nucleic acids research*, vol. 40, no. D1, pp. D1100–D1107, 2011.

[89] D. L. Mobley and J. P. Guthrie, “Freesolv: a database of experimental and calculated hydration free energies, with input files,” *Journal of computer-aided molecular design*, vol. 28, no. 7, pp. 711–720, 2014.

[90] G. W. Bemis and M. A. Murcko, “The properties of known drugs. 1. molecular frameworks,” *Journal of medicinal chemistry*, vol. 39, no. 15, pp. 2887–2893, 1996.

[91] J. Friedman, T. Hastie, R. Tibshirani, *et al.*, “Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors),” *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[92] S. Liu, M. F. Demirel, and Y. Liang, "N-gram graph: Simple unsupervised representation for graphs, with applications to molecules," in *NeurIPS*, 2019, pp. 8464–8476.

[93] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A. J. Smola, and Z. Zhang, "Deep graph library: Towards efficient and scalable deep learning on graphs," *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. [Online]. Available: https://arxiv.org/abs/1909.01315

[94] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[95] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICLR*, 2020.

[96] J. Saarikoski and M. Viluksela, "Influence of ph on the toxicity of substituted phenols to fish," *Archives of environmental contamination and toxicology*, vol. 10, no. 6, pp. 747–753, 1981.

[97] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[98] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[99] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang, "Self-supervised graph transformer on large-scale molecular data," *arXiv preprint arXiv:2007.02835*, 2020.

[100] Z. Xu, S. Wang, F. Zhu, and J. Huang, "Seq2seq fingerprint: An unsupervised deep molecular embedding for drug discovery," in *Proceedings of the 8th ACM*

international conference on bioinformatics, computational biology, and health informatics, 2017, pp. 285–294.

[101] S. Wang, Y. Guo, Y. Wang, H. Sun, and J. Huang, "Smiles-bert: large scale unsupervised pre-training for molecular property prediction," in *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, 2019, pp. 429–436.

[102] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," *arXiv preprint arXiv:1905.12265*, 2019.

[103] H. Ma, Y. Bian, Y. Rong, W. Huang, T. Xu, W. Xie, G. Ye, and J. Huang, "Multi-view graph neural networks for molecular property prediction," *arXiv preprint arXiv:2005.13607*, 2020.

[104] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[105] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra r-cnn: Towards balanced learning for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 821–830.

[106] B. Li, Y. Liu, and X. Wang, "Gradient harmonized single-stage detector," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8577–8584.

[107] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 1567–1578.

[108] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[109] C. Yan, Q. Ding, P. Zhao, S. Zheng, J. Yang, Y. Yu, and J. Huang, "Retroxpert: Decompose retrosynthesis prediction like a chemist," 2020.

[110] D. T. Jones, "Protein secondary structure prediction based on position-specific scoring matrices," *Journal of molecular biology*, vol. 292, no. 2, pp. 195–202, 1999.

[111] S. K. Sønderby and O. Winther, "Protein secondary structure prediction with long short term memory networks," *arXiv preprint arXiv:1412.7828*, 2014.

[112] N. Zhao, J. G. Han, C.-R. Shyu, and D. Korkin, "Determining effects of non-synonymous snps on protein-protein interactions using supervised and semi-supervised learning," *PLoS computational biology*, vol. 10, no. 5, p. e1003592, 2014.

[113] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 687–10 698.

[114] Y. Babakhin, A. Sanakoyeu, and H. Kitamura, "Semi-supervised segmentation of salt bodies in seismic images using an ensemble of convolutional neural networks," in *German Conference on Pattern Recognition.* Springer, 2019, pp. 218–231.

[115] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, "Rethinking pre-training and self-training," *Advances in neural information processing systems*, vol. 33, pp. 3833–3845, 2020.

[116] X. Li, Q. Sun, Y. Liu, Q. Zhou, S. Zheng, T.-S. Chua, and B. Schiele, "Learning to self-train for semi-supervised few-shot classification," *Advances in Neural Information Processing Systems*, vol. 32, pp. 10 276–10 286, 2019.

[117] Y. Cheng, "Semi-supervised learning for neural machine translation," in *Joint training for neural machine translation.* Springer, 2019, pp. 25–40.

[118] J. He, J. Gu, J. Shen, and M. Ranzato, "Revisiting self-training for neural sequence generation," *arXiv preprint arXiv:1909.13788*, 2019.

[119] A. Ghosh, H. Kumar, and P. Sastry, "Robust loss functions under label noise for deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[120] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[121] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, "Symmetric cross entropy for robust learning with noisy labels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 322–330.

[122] X. Ma, H. Huang, Y. Wang, S. Romano, S. Erfani, and J. Bailey, "Normalized loss functions for deep learning with noisy labels," in *International Conference on Machine Learning.* PMLR, 2020, pp. 6543–6553.

[123] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[124] V. G. Satorras, E. Hoogeboom, and M. Welling, "E (n) equivariant graph neural networks," *arXiv preprint arXiv:2102.09844*, 2021.

[125] L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17," *Journal of chemical information and modeling*, vol. 52, no. 11, pp. 2864–2875, 2012.

[126] J. A. DiMasi, H. G. Grabowski, and R. W. Hansen, “The cost of drug development,” *New England Journal of Medicine*, vol. 372, 2015.

[127] R. Berggren, M. Møller, R. Moss, P. Poda, and K. Smietana, “Outlook for the next 5 years in drug innovation,” *Nature reviews Drug discovery*, vol. 11, no. 6, pp. 435–436, 2012.

[128] S. Thakkar, T. Li, Z. Liu, L. Wu, R. Roberts, and W. Tong, “Drug-induced liver injury severity and toxicity (dilist): Binary classification of 1279 drugs by human hepatotoxicity,” *Drug Discovery Today*, vol. 25, no. 1, pp. 201–208, 2020.

[129] D. A. Parasrampuria, L. Z. Benet, and A. Sharma, “Why drugs fail in late stages of development: case study analyses from the last decade and recommendations,” *The AAPS Journal*, vol. 20, no. 3, p. 46, 2018.

[130] G. A. Kullak-Ublick, R. J. Andrade, M. Merz, P. End, A. Benesic, A. L. Gerbes, and G. P. Aithal, “Drug-induced liver injury: recent advances in diagnosis and risk assessment,” *Gut*, vol. 66, no. 6, pp. 1154–1164, 2017.

[131] H. Olson, G. Betton, D. Robinson, K. Thomas, A. Monro, G. Kolaja, P. Lilly, J. Sanders, G. Sipes, W. Bracken, *et al.*, “Concordance of the toxicity of pharmaceuticals in humans and in animals,” *Regulatory Toxicology and Pharmacology*, vol. 32, no. 1, pp. 56–67, 2000.

[132] N. Greene, L. Fisk, R. T. Naven, R. R. Note, M. L. Patel, and D. J. Pelletier, “Developing structureactivity relationships for the prediction of hepatotoxicity,” *Chemical Research in Toxicology*, vol. 23, no. 7, pp. 1215–1222, 2010.

[133] X. Zhu and N. L. Kruhlak, “Construction and analysis of a human hepatotoxicity database suitable for qsar modeling using post-market safety data,” *Toxicology*, vol. 321, pp. 62–72, 2014.

[134] J. J. Xu, P. V. Henstock, M. C. Dunn, A. R. Smith, J. R. Chabot, and D. de Graaf, "Cellular imaging predictions of clinical drug-induced liver injury," *Toxicological Sciences*, vol. 105, no. 1, pp. 97–105, 2008.

[135] M. Z. Sakatis, M. J. Reese, A. W. Harrell, M. A. Taylor, I. A. Baines, L. Chen, J. C. Bloomer, E. Y. Yang, H. M. Ellens, J. L. Ambroso, *et al.*, "Preclinical strategy to reduce clinical hepatotoxicity using in vitro bioactivation data for ¿200 compounds," *Chemical Research in Toxicology*, vol. 25, no. 10, pp. 2067–2082, 2012.

[136] M. Chen, A. Suzuki, S. Thakkar, K. Yu, C. Hu, and W. Tong, "Dilirank: the largest reference drug list ranked by the risk for developing drug-induced liver injury in humans," *Drug Discovery Today*, vol. 21, no. 4, pp. 648–653, 2016.

[137] M. Chen, V. Vijay, Q. Shi, Z. Liu, H. Fang, and W. Tong, "Fda-approved drug labeling for the study of drug-induced liver injury," *Drug Discovery Today*, vol. 16, no. 15-16, pp. 697–703, 2011.

[138] E. Minerali, D. H. Foil, K. M. Zorn, T. R. Lane, and S. Ekins, "Comparing machine learning algorithms for predicting drug-induced liver injury (dili)," *Molecular Pharmaceutics*, vol. 17, no. 7, pp. 2628–2637, 2020.

[139] M. D. Aleo, F. Shah, S. Allen, H. A. Barton, C. Costales, S. Lazzaro, L. Leung, A. Nilson, R. S. Obach, A. D. Rodrigues, *et al.*, "Moving beyond binary predictions of human drug-induced liver injury (dili) toward contrasting relative risk potential," *Chemical Research in Toxicology*, vol. 33, no. 1, pp. 223–238, 2020.

[140] J. R. Mora, Y. Marrero-Ponce, C. R. García-Jacas, and A. Suarez Causado, "Ensemble models based on qubils-mas features and shallow learning for the prediction of drug-induced liver toxicity: Improving deep learning and traditional approaches," *Chemical Research in Toxicology*, vol. 33, no. 7, pp. 1855–1873, 2020.

[141] R. Ancuceanu, M. V. Hovanet, A. I. Anghel, F. Furtunescu, M. Neagu, C. Constantin, and M. Dinu, "Computational models using multiple machine learning algorithms for predicting drug hepatotoxicity with the dilirank dataset," *International Journal of Molecular Sciences*, vol. 21, no. 6, p. 2114, 2020.

[142] N. M. O'Boyle, C. M. Campbell, and G. R. Hutchison, "Computational design and selection of optimal organic photovoltaic materials," *The Journal of Physical Chemistry C*, vol. 115, no. 32, pp. 16 200–16 210, 2011.

[143] T. Le, R. Winter, F. Noe, and D.-A. Clevert, "Neuraldecipher - reverse-engineering ecfp fingerprints to their molecular structures," 5 2020.

[144] H. Ma, Y. Rong, W. Huang, T. Xu, W. Xie, G. Ye, and J. Huang, "Multi-view graph neural networks for molecular property prediction," *arXiv preprint arXiv:2005.13607*, 2020.

[145] S. A. Shahane, R. Huang, D. Gerhold, U. Baxa, C. P. Austin, and M. Xia, "Detection of phospholipidosis induction: A cell-based assay in high-throughput and high-content format," *Journal of Biomolecular Screening*, vol. 19, no. 1, pp. 66–76, 2014.

[146] S. Sun, "A survey of multi-view machine learning," *Neural computing and applications*, vol. 23, no. 7-8, pp. 2031–2038, 2013.

[147] H. Sun, M. Xia, S. A. Shahane, A. Jadhav, C. P. Austin, and R. Huang, "Are herg channel blockers also phospholipidosis inducers?" *Bioorganic & Medicinal Chemistry Letters*, vol. 23, no. 16, pp. 4587–4590, 2013.

[148] M. Xia, S. A. Shahane, R. Huang, S. A. Titus, E. Shum, Y. Zhao, N. Southall, W. Zheng, K. L. Witt, R. R. Tice, *et al.*, "Identification of quaternary ammonium compounds as potent inhibitors of herg potassium channels," *Toxicology and Applied Pharmacology*, vol. 252, no. 3, pp. 250–258, 2011.

[149] J. Kazius, R. McGuire, and R. Bursi, “Derivation and validation of toxicophores for mutagenicity prediction,” *Journal of Medicinal Chemistry*, vol. 48, no. 1, pp. 312–320, 2005.

[150] K. Hansen, S. Mika, T. Schroeter, A. Sutter, A. Ter Laak, T. Steger-Hartmann, N. Heinrich, and K.-R. Muller, “Benchmark data set for in silico prediction of ames mutagenicity,” *Journal of Chemical Information and Modeling*, vol. 49, no. 9, pp. 2077–2081, 2009.

[151] M. S. Attene-Ramos, R. Huang, S. Michael, K. L. Witt, A. Richard, R. R. Tice, A. Simeonov, C. P. Austin, and M. Xia, “Profiling of the tox21 chemical collection for mitochondrial function to identify compounds that acutely decrease mitochondrial membrane potential,” *Environmental Health Perspectives*, vol. 123, no. 1, pp. 49–56, 2015.

[152] M. S. Attene-Ramos, R. Huang, S. Sakamuru, K. L. Witt, G. C. Beeson, L. Shou, R. G. Schnellmann, C. C. Beeson, R. R. Tice, C. P. Austin, *et al.*, “Systematic study of mitochondrial toxicity of environmental chemicals using quantitative high throughput screening,” *Chemical Research in Toxicology*, vol. 26, no. 9, pp. 1323–1332, 2013.

[153] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2011.

[154] J. Turian, L. Ratinov, and Y. Bengio, “Word representations: A simple and general method for semi-supervised learning,” in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ser. ACL ’10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 384–394. [Online]. Available: http://dl.acm.org/citation.cfm?id=1858681.1858721

[155] Z.-H. Zhou and J.-M. Xu, "On the relation between multi-instance learning and semi-supervised learning," in *Proceedings of the 24th international conference on Machine learning.* ACM, 2007, pp. 1167–1174.

[156] P. Liang, "Semi-supervised learning for natural language," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[157] A. Søgaard, "Semi-supervised learning and domain adaptation in natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 6, no. 2, pp. 1–103, 2013.

[158] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning.* ACM, 2008, pp. 160–167.

[159] Y. Hu, E. Lounkine, and J. Bajorath, "Improving the search performance of extended connectivity fingerprints through activity-oriented feature filtering and application of a bit-density-dependent similarity function," *ChemMedChem*, vol. 4, no. 4, pp. 540–548, 2009.

[160] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[161] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[162] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in neural information processing systems*, 1997, pp. 473–479.

[163] T. Sterling and J. J. Irwin, "Zinc 15–ligand discovery for everyone," *Journal of chemical information and modeling*, vol. 55, no. 11, pp. 2324–2337, 2015.

[164] B. Ramsundar, P. Eastman, P. Walters, V. Pande, K. Leswing, and Z. Wu, *Deep Learning for the Life Sciences.* O'Reilly Media, 2019, https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

[165] R. Li and J. Huang, "Learning graph while training: An evolving graph convolutional neural network," *arXiv preprint arXiv:1708.04675*, 2017.

BIOGRAPHICAL STATEMENT

Hehuan Ma received her Ph.D. in Computer Science and Engineering from the University of Texas at Arlington at 2023. Prior to beginning the Ph.D. program, Hehuan obtained her M.S. degree from Louisiana Tech University, and B.S. degree from Huaqiao University, China. Her main research mainly lies in the areas of deep learning, graph neural networks, and their applications in drug discovery, especially molecular property prediction. During her Ph.D. study, she has published more than 10 conference and journal papers, such as Bioinformatics, ACM Conference on Bioinformatics, Computational Biology and Biomedicine (ACM BCB), IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Chemical research in toxicology, International Conference on Computer Vision (ICCV), AAAI Conference on Artificial Intelligence (AAAI), International Conference on Research in Computational Molecular Biology (RECOMB), Journal of Computational Biology (JCB), Biomolecules, and Frontiers in Big Data. She has been invited to serve as a reviewer for many top-tier conferences and journals, such as KDD, ICML, NeurIPS, IJCAI, AAAI, MICCAI, IEEE Transactions on Neural Networks and Learning Systems (TNNLS), Chemical research in toxicology.