

Few-Shot Graph Learning for Molecular Property Prediction

Zhichun Guo¹, Chuxu Zhang^{2*}, Wenhao Yu¹

John Herr¹, Olaf Wiest¹, Meng Jiang¹, Nitesh V. Chawla^{1*}

¹University of Notre Dame, IN, USA ²Brandeis University, MA, USA

zguo5@nd.edu, chuxuzhang@brandeis.edu, {wyu1, jherr1, Olaf.G.Wiest.1, mjiang2, nchawla}@nd.edu

ABSTRACT

The recent success of graph neural networks has significantly boosted molecular property prediction, advancing activities such as drug discovery. The existing deep neural network methods usually require large training dataset for each property, impairing their performance in cases (especially for new molecular properties) with a limited amount of experimental data, which are common in real situations. To this end, we propose Meta-MGNN, a novel model for few-shot molecular property prediction. Meta-MGNN applies molecular graph neural network to learn molecular representations and builds a meta-learning framework for model optimization. To exploit unlabeled molecular information and address task heterogeneity of different molecular properties, Meta-MGNN further incorporates molecular structures, attribute based self-supervised modules and self-attentive task weights into the former framework, strengthening the whole learning model. Extensive experiments on two public multi-property datasets demonstrate that Meta-MGNN outperforms a variety of state-of-the-art methods.

KEYWORDS

Molecular Property Prediction, Few-Shot Learning, Graph Learning

ACM Reference Format:

Zhichun Guo¹, Chuxu Zhang^{2*}, Wenhao Yu¹, John Herr¹, Olaf Wiest¹, Meng Jiang¹, Nitesh V. Chawla¹. 2021. Few-Shot Graph Learning for Molecular Property Prediction. In *Proceedings of the Web Conference 2021 (WWW '21)*, April 19–23, 2021, Ljubljana, Slovenia. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3442381.3450112>

1 INTRODUCTION

Drug discovery significantly benefits all human beings, especially for public health during this tough and special time caused by COVID-19 [26]. Developing and discovering new drugs is a time, resource, and money consuming process. A key step is to test a large number of molecules for therapeutic activity through extensive biological studies [23]. Unfortunately, these discovered ones often fail to become the approved drug candidates for various reasons such as low activity or toxicity [30]. Researchers need to select a great number of similar molecules as potential candidates. To find the molecules which have the same efficacious property, these

selected molecules need to be tested through a complex experimental process. After that, only a few or even no molecules will be remaining as possible drug candidates to be tested further for risk and pharmaceutical activity. Therefore, it is crucial to improve the effectiveness of filtering the most likely drug candidates before taking experiments via wet-lab experimentation, thus wasting less time and resources on molecules that are unlikely to proceed to the lead stage. This concept is generally described as "fail early-fail cheap".

Virtual screening is a widely used approach to screen out molecules likely to fail early, which avoids a large set of molecules to be investigated [22, 23]. Recent advances in deep learning have played an important role in virtual screening. These deep learning techniques have inspired novel approaches to a better understanding of molecules and their properties through molecular representation learning [7, 10, 35, 43, 45]. Deep neural networks learn more about specific molecular properties when they are fed with more instances during training. Thus, deep learning models require a large amount of training data to achieve desired capability and satisfactory performance [3]. However, it is common that there are only a few known molecules that share the same set of properties [1, 30]. We analyzed the datasets in MoleculeNet [33], a well-known benchmark for predicting molecular properties. We find that more than half of the properties are shared by fewer than 100 molecules across several datasets. This is a case of the well-known problem of *few-shot* available data, which seriously impairs the performances of current approaches. Therefore, it is essential to develop a deep neural model for predicting molecular properties effectively in few-shot scenarios.

There are several challenges that need to be overcome to achieve this goal. Molecules can be considered as a heterogeneous structure where each atom connects to different neighboring atoms via different types of bonds. Previous work [29] represents molecules as *SMILES* strings and leverages sequence models [21, 29] to learn molecular embedding. This approach is not able to capture information in each bond well [32]. This is because bonds in molecules not only represent connected relations between different atoms but also contain attributed information that characterizes the bond type such as single, double, or triple. Thus, the first challenge is to design a deep neural network that can discover *effective* molecular representations from *few-shot* data. Because only a limited amount of labeled molecular property data are available, the second challenge is to exploit the useful *unlabeled* information in molecule data and further develop an *efficient* learning procedure to transfer the knowledge from other property prediction, so that the model can fast adapt to the novel (new) molecular properties with limited data. Moreover, different molecular properties could represent quite different molecular structures. Thus, their data should be treated differently in the knowledge transfer process. The third challenge

*Corresponding authors

This paper is published under the Creative Commons Attribution 4.0 International (CC-BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

© 2021 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC-BY 4.0 License.

ACM ISBN 978-1-4503-8312-7/21/04.

<https://doi.org/10.1145/3442381.3450112>

is to distinguish the *different* importance of molecular properties when performing the efficient learning procedure.

To address the above challenges, we propose a novel model called Meta-MGNN for few-shot molecular property prediction. First, we leverage graph neural network with the pre-training process to fuse heterogeneous molecular graph information as molecular embedding. Then, we develop a meta-learning framework to transfer knowledge from different property prediction tasks and obtain a well-initialized model which could be fast adapted to a new molecular property with limited data. In order to exploit and capture unlabeled information in molecule data, we design a self-supervised module which consists of a bond reconstruction loss and an atom type prediction loss, accompanied by the main property prediction loss. Moreover, considering different property prediction tasks contribute differently to the few-shot learner, we further introduce a self-attentive task weight to measure their importance. Both self-supervised module and self-attentive task weight are incorporated into the meta-learning procedure for strengthening the model.

Contributions. To summarize, the main contributions of this work are as follows:

- We formulate the molecular property prediction as a few-shot learning problem, which exploits the rich information in various properties to address the lack of laboratory data problem for each individual property.
- To deal with the few-shot challenge, we propose a novel model called Meta-MGNN by exploring graph neural network, self-supervised learning, and task weight aware meta-learning.
- We conduct extensive experiments on two public datasets and the evaluation results demonstrate the superior performance of Meta-MGNN over state-of-the-art methods. The effectiveness of each model component is also verified.

2 RELATED WORK

In this section, we review existing work including graph neural network, few-shot learning, and molecular property prediction.

Graph Neural Network (GNN). GNNs have gained increasing popularity due to its capability of modeling graph-structured data [9, 27, 39]. Typically, a GNN model uses a neighborhood aggregation function to iteratively update the representation of a node by aggregating representations of its neighboring nodes and edges. GNNs have showed attractive performance in various applications, such as recommendation systems [4, 24], behavior modeling [38], and anomaly detection [44]. Molecular property prediction is also a popular application of GNNs since a molecule could be represented as a topological graph by treating atoms as nodes, and bonds as edges [7, 8, 18, 20]. We will elaborate them in the next paragraph.

Molecular Property Prediction. Methods can be categorized into two main groups based on the input molecular type: (1) molecular graph, and (2) simplified molecular-input line-entry (SMILES) [31]. For the first group of methods, each molecule is represented as a graph associated with different atom nodes interconnected by bond edges. One typical way is to employ graph neural networks to learn molecular representations [7, 8, 10, 18, 20]. For example, Lu et al. [18] proposed a novel hierarchical GNN. It includes an embedding layer, a Radial Basis Function layer, and an interaction

layer to learn molecular representations from different levels. Hu et al. [10] proposed several novel pre-training strategies to pre-train GNNs at the level of individual nodes and the entire graph to learn local and global molecular representations simultaneously. For the second type of representation, SMILES is a sequence notation for describing the structure of molecules. Researchers take molecules as sequences and adopt language models to learn their representations [29, 43, 45]. For example, Zhang et al. [43] proposed a semi-supervised Seq2Seq fingerprint model which contains three ends of one input, one supervised output, and one unsupervised output. Zheng et al. [45] presented a new model to study structure-property relationships through a self-attentive linear notation syntax analysis. Guo et al. [8] proposed a novel graph and sequence fusion learning model to capture information both from the molecular graph structure and SMILES. Here, we take each molecule as a graph as it preserves the molecular inner structure better, and employ graph neural networks to learn their representations.

Few-shot Learning. Successes of few-shot learning have been accomplished in various application domains such as computer vision [5, 11] and graph learning [6, 36, 37, 40, 41]. There are two notable types of few-shot learning approaches: (1) *metric-based learning* and (2) *gradient-based learning*. The former learns a generative metric to compare and match few-examples [2, 25, 28]. Vinyals et al. [28] proposed a novel matching metric, named Matching Nets, to match unlabeled examples to the class of few-shot labeled examples. Sung et al. [25] proposed relation network which learns a deep distance metric to compute relation scores of different images and further classify images. The latter aims to employ a specific meta-learner to learn well-initialized parameters of the base model for different tasks [5, 15, 42]. For instance, Finn et al. [5] proposed MAML which designs this kind of meta-learner to effectively initialize a base-learner that could be fast adapted to new tasks. In this work, our few-shot learning strategy is gradient-based learning.

3 PRELIMINARY

In this section, we first define the few-shot molecular property prediction problem, then present the details of using graph neural network (GNN) for learning molecular representations.

3.1 Problem Definition

Let $G = (\mathcal{V}, \mathcal{E})$ denote a molecular graph where \mathcal{V} is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. Particularly, a node in a molecular graph represents a chemical atom and an edge represents a chemical bond between two atoms. Given a set of molecular graphs $\mathcal{G} = \{G_1, \dots, G_N\}$ and their labels $\mathcal{Y} = \{y_1, \dots, y_N\}$, the goal of molecular property prediction is to learn a molecular representation vector for predicting its label (i.e., molecular property) of each $G_i \in \mathcal{G}$, i.e., to learn a mapping function $f_\theta : \mathcal{G} \rightarrow \mathcal{Y}$.

Unlike previous studies where there are enough examples for each new property prediction task, this work considers a more practical scenario that only few-shot samples are given. Specifically, we aim to develop a classifier which can be fast adapted to predict new molecular properties that are *unseen* during the training process, given only a few samples of these new properties. Formally, the problem is defined as follows.

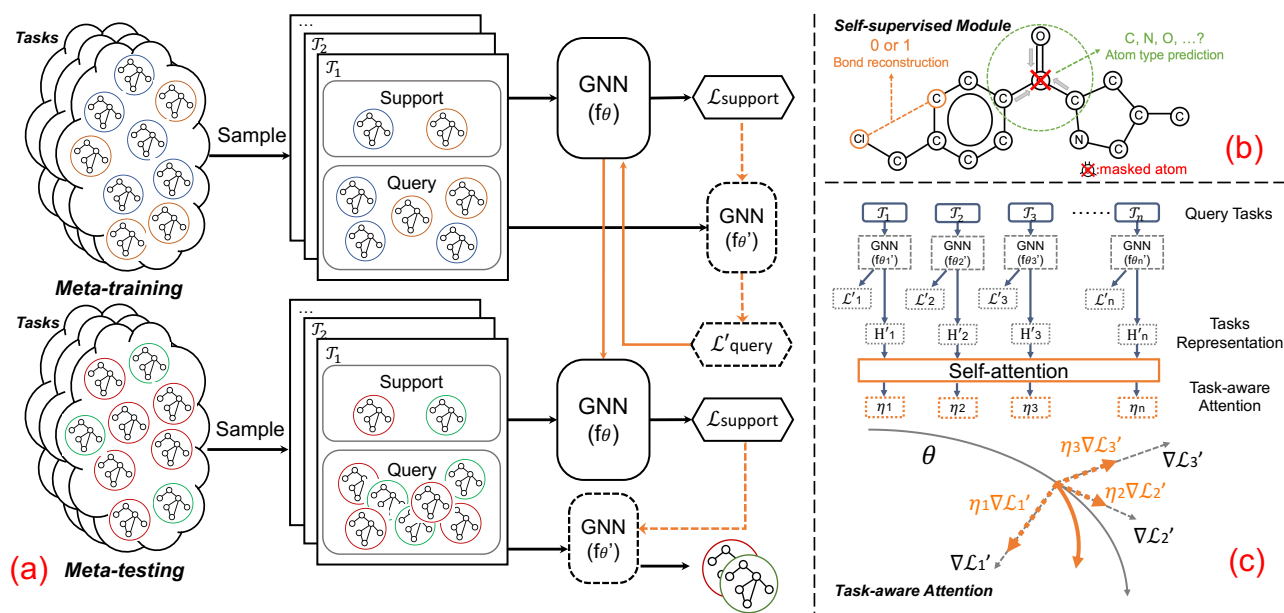


Figure 1: (a) The overall framework of Meta-MGNN: It first samples a batch of training tasks. For each task, there are a few data examples in the support set. These examples are fed into a GNN parameterized by θ . Then the support loss $\mathcal{L}_{support}$ is calculated and utilized to update the GNN parameters to θ' . Next, the examples in the corresponding query set are fed into the GNN parameterized by θ' and calculate the loss \mathcal{L}'_{query} for this task. The same process repeats for other training tasks. Later, we compute the summation of \mathcal{L}'_{query} over all sampled tasks and use it to further update the GNN parameters for testing. (b) Self-supervised module: It includes bond reconstruction and atom type prediction. The orange part shows that we sample two atoms and use GNN to predict if there is a bond between them. The green part shows that we mask several atoms randomly and use GNN to predict their types. (c) Task-aware attention: It calculates the average of all the molecular embedding from the query set of the same task to represent this task. With the embedding of each task, we design a self-attentive layer to compute the weight of each task, then incorporate it into a meta-training process for updating model parameters θ .

PROBLEM 1. Few-Shot Molecular Property Prediction Given molecular properties $\mathcal{Y} = \{y_1, \dots, y_N\}$ and their corresponding few-shot molecular graph sets $\{\mathcal{G}_1 \in y_1, \dots, \mathcal{G}_N \in y_N\}$ (training data), the task is to design a machine learning model to predict molecular graphs of new properties that only have few-shot examples (test data).

3.2 Molecular Graph Neural Network

By viewing molecular structure as graph data (i.e., molecular graph), recent deep learning methods for graphs, such as graph neural networks (GNNs) [32, 39], can be utilized to learn molecular representations which are fed to downstream machine learning models for molecular property prediction [17]. In this section, we will present the details of employing GNNs to obtain molecular representations.

A GNN model is able to utilize both graph structure and node/edge features information to learn a representation vector \mathbf{h}_v for each node $v \in \mathcal{V}$. Specifically, a GNN model uses a neighborhood aggregation function to iteratively update the representation of a node by aggregating representations of its neighboring nodes and edges. After l iterations, a node representation $\mathbf{h}_v^{(l)}$ is able to capture the information within its l -hop neighborhoods. In a molecular graph, each node represents an atom and each edge represents a chemical

bond between two atoms. As the input layer of GNN, we first initialize representations of both nodes and edges using their attributes in molecular graph. The node attributes include atom number (AN) and chirality tag (CT), and edge attributes include bond type (BT) and bond direction (BD). Formally, we initialize node representation as $\mathbf{h}_v^{(0)} = \mathbf{v}_{AN} \oplus \mathbf{v}_{CT}$ and edge representation as $\mathbf{h}_e^{(0)} = \mathbf{e}_{BT} \oplus \mathbf{e}_{BD}$, where \mathbf{v} and \mathbf{e} denote node/edge attributes and \oplus is concatenation operator. Then, the node representation $\mathbf{h}_v^{(l)}$ at the l -th layer of GNN is formulated as:

$$\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \text{AGG}_l(\{\mathbf{h}_u^{(l-1)} : \forall u \in \mathcal{N}(v)\}, \{\mathbf{h}_e^{(l-1)} : e = (v, u)\}), \quad (1)$$

$$\mathbf{h}_v^{(l)} = \sigma(\mathbf{W}^{(l)} \cdot \text{CONCAT}(\mathbf{h}_v^{(l-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(l)})), \quad (2)$$

where $\mathcal{N}(v)$ is the neighbor set of v , $\sigma(\cdot)$ is a non-linear activation function (e.g., LeakyReLU). $\text{AGG}(\cdot)$ is an aggregating function. A number of architectures for $\text{AGG}(\cdot)$ have been proposed in recent years such as graph convolutional neural network (GCN) [12] and graph attention network (GAT) [27]. Here, we use graph isomorphism network (GIN) [34], which has demonstrated state-of-the-art performance on a variety of benchmark tasks. After that, we can learn the representation of each node in molecular graph:

$\mathbf{h}_v = \mathbf{h}_v^{(l)} / \|\mathbf{h}_v^{(l)}\|_2$. To obtain the graph-level representation \mathbf{h}_G for a molecular graph, we calculate the average node embeddings at the final layer:

$$\mathbf{h}_G = \text{MEAN}(\{\mathbf{h}_v^{(l)} : v \in \mathcal{V}\}), \quad (3)$$

The graph-level molecular representation \mathbf{h}_G can be further fed into a classifier (e.g., a multi-layer perception) for molecular property prediction, as we will present in the next section.

Pre-trained Molecular Graph Neural Network. Pre-trained models have been widely used in natural language processing, computer vision, and graph analysis in recent years [3, 10]. In general, pre-training allows a model to learn universal representations, provides a better parameter initialization, and avoids overfitting on downstream tasks with small training data. Models with pre-training have been demonstrated to obtain superior performance than models without that. Therefore, we are motivated to leverage the recent pre-trained graph neural network technique (PreGNN) [10] to obtain parameter initialization of molecular graph neural network.

4 META-MGNN

In this section, we present the details of proposed the Meta-MGNN for few-shot molecular property prediction. Meta-MGNN is built on MGNN and employs a meta-learning framework for model initialization and adaption. Molecular structure and feature based self-supervised module and self-attentive task weight are further incorporated into the former framework for model enhancement.

4.1 Meta-learning Setup

We build the meta-learning framework based on MAML [5]. Given the model f_θ with learnable parameters θ that maps molecular graph to specific properties such as toxicity, i.e., $f_\theta : \mathcal{G} \rightarrow \mathcal{Y}$. In meta-learning, the model is expected to adapt to a number of different tasks, i.e., predicting different kinds of molecular properties. Particular, in the k -shot meta-learning, for each task \mathcal{T}_τ sampled from distribution $p(\mathcal{T})$, the model is trained using only k data samples and further tested on remaining data samples of \mathcal{T}_τ . In this setting, we refer to the corresponding training and test sets of each task as *support* set and *query* set, denoted as $\mathcal{T}_\tau = \{\mathcal{G}_\tau, \mathcal{Y}_\tau, \mathcal{G}'_\tau, \mathcal{Y}'_\tau\}$, where $\mathcal{G}_\tau, \mathcal{Y}_\tau$ are *support* sets of input molecular graphs and property labels, and $\mathcal{G}'_\tau, \mathcal{Y}'_\tau$ are *query* sets of input molecular graphs and property labels. During meta-training, the model f_θ is first updated to task-specific model using *support* set of each task, then further optimized to task-agnostic model using prediction loss over the *query* set of all tasks in training data. After sufficient training, the learned model can be further utilized to predict new tasks (new molecular properties) with only k data samples as support set, which is called meta-testing. To avoid data overlapping, data of tasks used for meta-testing are held out during meta-training. The whole framework is illustrated in Figure 1(a).

4.2 Meta-training

In meta-training, the goal is to obtain well initialized model parameters θ that can be generally applicable to different tasks, and explicitly encourage the initialized parameters to perform well after a small number of gradient descent updates on a new task with few-shot data. When adapting to a task \mathcal{T}_τ , we begin with feeding

Algorithm 1: Meta-MGNN

Require: $\{\mathcal{G}_\tau, \mathcal{Y}_\tau\}$: support data ; $\{\mathcal{G}'_\tau, \mathcal{Y}'_\tau\}$: query data; α , β : step sizes (i.e., learning rates)

```

1  $\theta \leftarrow$  Pre-trained by PreGNN [10]
2 while not done do
3   Sample batch of tasks  $\mathcal{T}_\tau \sim p(\mathcal{T})$ 
4   for all  $\mathcal{T}_\tau$  do
5     Sample  $k$  examples  $\{G_{\tau 1}, G_{\tau 2}, \dots, G_{\tau k}\} \in \mathcal{G}_\tau$ 
6     for  $i=1$  to  $k$  do
7        $y_{\tau i}, \mathbf{h}_{\tau i} = \text{GNN}(G_{\tau i}, \theta)$ 
8     end
9      $\mathbf{H}_\tau = \text{MEAN}(\mathbf{h}_{\tau 1}, \mathbf{h}_{\tau 2}, \dots, \mathbf{h}_{\tau k})$ 
10     $\mathcal{L}_\tau \leftarrow$  Eq. (9) with  $\{y_{\tau 1}, y_{\tau 2}, \dots, y_{\tau k}\}$ 
11     $\theta'_\tau = \theta - \alpha \nabla \mathcal{L}_\tau$ 
12    Sample  $n$  examples  $\{G'_{\tau 1}, G'_{\tau 2}, \dots, G'_{\tau n}\} \in \mathcal{G}'_\tau$ 
13    for  $j=1$  to  $n$  do
14       $y'_{\tau j}, \mathbf{h}'_{\tau j} = \text{GNN}(G'_{\tau j}, \theta'_\tau)$ 
15    end
16     $\mathcal{L}'_\tau \leftarrow$  Eq. (9) with  $\{y'_{\tau 1}, y'_{\tau 2}, \dots, y'_{\tau n}\}$ 
17  end
18   $\{\eta(\mathcal{T}_1), \dots, \eta(\mathcal{T}_l)\} \leftarrow$  Eq. (11) with  $\{\mathbf{H}_1, \dots, \mathbf{H}_l\}$ 
19   $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_\tau \sim p(\mathcal{T})} \eta(\mathcal{T}_\tau) \cdot \mathcal{L}'_\tau$ 
20 end
```

the *support* set to the model and calculate the loss $\mathcal{L}_{\mathcal{T}_\tau}$ to update parameters θ to θ'_τ through gradient descent:

$$\theta'_\tau = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_\tau}(\theta), \quad (4)$$

where α is the step size. It should be noted that Eq.(4) only shows one-step gradient update while we can take multiple-steps gradient update in practice.

4.2.1 Loss Function. Typically, the above loss $\mathcal{L}_{\mathcal{T}_\tau}$ is calculated by the supervised signals from downstream tasks [5, 46], i.e., molecular property labels in this study. However, simply using supervised signals maybe not effective since only a few samples are given for each task. In addition, the complexity of molecules inherently bring useful unlabeled information in both structure and attribute. Therefore, to enhance the above meta-training process, we propose to exploit and leverage unlabeled information in molecular graphs. In particular, we design a self-supervised module which consists of a *bond reconstruction loss* and an *atom type prediction loss*, accompanying with the property prediction loss.

Molecular Property Prediction Loss. To predict molecular property, we introduce a multi-layer perception (MLP) on top of the graph-level molecular representation \mathbf{h} (Eq.(3)), i.e., $\hat{y} = \text{MLP}(\mathbf{h})$. The loss of prediction is defined as the *cross entropy* loss between the predicted labels and ground-truth labels:

$$\mathcal{L}_{\text{label}}(\theta) = -\frac{1}{k} \sum_{i=1}^k \text{CROSSENTROPY}(y_i, \hat{y}_i) \quad (5)$$

where k is the number of data samples.

Bond Reconstruction Loss. To perform bond reconstruction in molecular graphs, we first sample a set of positive edges (existing

bonds) in the molecular graph, then sample a set of negative edges (non-existing bonds) by choosing node pairs that do not have an edge in the original molecular graph. We denote \mathcal{E}_s as the union set of sampled positive edges and negative edges. In practice, we set $|\mathcal{E}_s| = 10$ including 5 positive samples and 5 negative samples. The bond reconstruction score is computed by the inner product of embeddings between the sampled pair of nodes, i.e., $\hat{e}_{uv} = \mathbf{h}_v^\top \cdot \mathbf{h}_u$. The bond reconstruction loss is defined as the binary cross entropy loss between the predicted bonds and ground-truth bonds:

$$\mathcal{L}_{edge}(\theta) = -\frac{1}{|\mathcal{E}_s|} \sum_{e_{uv} \in \mathcal{E}_s} \text{BINARYCROSSENTROPY}(e_{uv}, \hat{e}_{uv}) \quad (6)$$

Atom Type Prediction Loss. In a molecule, different atoms are connected in a certain way (e.g., carbon-carbon bond, carbon-oxygen bond), leading to different molecular structure. The atom type determines how a node in the molecular graph connects with neighboring nodes. Thus, we utilize the contextual sub-graph of a node (atom) to predict its type. Specifically, we first sample a set of nodes in a molecular graph, denoted as $\mathcal{V}_{ct} \subseteq \mathcal{V}$. For each node v in \mathcal{V}_{ct} , the contextual sub-graph is defined as its neighbors within l -hops, i.e., $G_{sub} = (\mathcal{U}_{sub}, \mathcal{E}_{sub})$ where $\mathcal{U}_{sub} = \{v\} \cup \mathcal{N}_l(v)$, $\mathcal{E}_{sub} \subseteq \mathcal{U}_{sub} \times \mathcal{U}_{sub}$, and $\mathcal{N}_l(v)$ represents the set of neighboring nodes of node v . In practice, we choose $\mathcal{V}_{ct} = 15\%$ nodes in the graph and $l = 1$. Later, we use a multi-layer perception (MLP) on the top of mean pooling of all nodes in the contextual sub-graph excluding the central node and the atom type prediction loss is formulated as the cross entropy loss between predicted node type and ground-truth node type:

$$\hat{v}_i = \text{MLP}(\text{MEAN}(\{\mathbf{h}_u : u \in \mathcal{N}_l(v)\})), \quad (7)$$

$$\mathcal{L}_{node}(\theta) = -\frac{1}{|\mathcal{V}_c|} \sum_{i=1}^{|\mathcal{V}_c|} \text{CROSSENTROPY}(v_i, \hat{v}_i), \quad (8)$$

The self-supervised module (of both bond reconstruction and atom type prediction) is illustrated in Figure 1(b).

Joint Loss. The loss for task \mathcal{T}_τ in the meta-training process is formulated as the summation over the above three losses,

$$\mathcal{L}_{\mathcal{T}_\tau}(\theta) = \mathcal{L}_{node}(\theta) + \lambda_1 \mathcal{L}_{edge}(\theta) + \lambda_2 \mathcal{L}_{label}(\theta) \quad (9)$$

where λ_1 and λ_2 are trade-off parameters that control the importance of different losses. In practice, we set $\lambda_1 = \lambda_2 = 0.1$.

4.2.2 Task-aware Attention. With the new model parameter θ'_τ obtained from the support set data of task \mathcal{T}_τ , i.e., $\theta'_\tau = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_\tau}(\theta)$, the model is further updated as follows:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_\tau \sim p(\mathcal{T})} \eta(\mathcal{T}_\tau) \cdot \mathcal{L}'_{\mathcal{T}_\tau}(\theta'_\tau), \quad (10)$$

where β is meta-learning rate, $\mathcal{L}'_{\mathcal{T}_\tau}$ is the joint loss over *query* set of \mathcal{T}_τ . In other words, the model parameters θ are further updated over losses of all sampled tasks through gradient descent. The traditional meta-learning methods (e.g., MAML [5]) treat each task with the same weight when optimizing the meta-learner (i.e., $\eta(\mathcal{T}_\tau)$ are same for all tasks), which cannot reflect how important of different property prediction tasks are. Therefore, considering different property prediction tasks contribute differently to the meta-learner optimization, we further introduce a self-attentive weight to measure task

Table 1: Details of atom and bond features.

# Atom Type	118
Atom Chirality Tag	Unspecified, Tetrahedral cw, Tetrahedral ccw, Other
Bond Type	Single, Double, Triple, Aromatic
Bond Direction	-, Endupright, Enddownright

importance. Particular, we use self-attentive mechanism [16] to calculate importance of each task:

$$\eta(\mathcal{T}_\tau) = \frac{\exp(\text{MLP}(\mathbf{H}_{\mathcal{T}_\tau}))}{\sum_{\mathcal{T}_{\tau'} \in \mathcal{T} \exp(\text{MLP}(\mathbf{H}_{\mathcal{T}_{\tau'}}))}, \mathbf{H}_{\mathcal{T}_\tau} = \text{MEAN}(\{\mathbf{h}_{\mathcal{T}_\tau, i}\}_{i=1}^k). \quad (11)$$

where \mathcal{T} is the set of all tasks, $\mathbf{H}_{\mathcal{T}_\tau}$ denotes the task embedding which is computed by averaging all molecular embeddings of \mathcal{T}_τ . Figure 1(c) illustrates the details of self-attentive task weight computation. The meta-training process is described in Algorithm 1.

4.2.3 Meta-testing. During meta-testing, we first utilize the few-shot *support* set of new tasks to update parameters θ of Meta-MGNN via one or a small number of gradient descent steps using Eq. (4), then evaluate performance in *query* set.

5 EXPERIMENTS

In this section, we conduct extensive experiments on two public datasets (Tox21 and Sider) to compare performances of different models and show related analysis.

5.1 Datasets

We evaluate different methods on the Tox21 and Sider datasets. They are collected from MoleculeNet [33], which is a large scale benchmark dataset for molecular machine learning. Tox21 has 7,831 instances with 12 different tasks; Sider has 1,427 instances with 27 different tasks. In each task, molecules are divided into positive instances and negative instances (i.e., binary labels). A positive instance means that a molecule has a specific property, and a negative instance means that a molecule does not have the property. We manually split 3 tasks from Tox21 and 6 tasks from Sider for meta-testing. The details of these two datasets are as follows:

- **Tox21**¹: Toxicity on 12 biological targets, including nuclear receptors and stress response pathways.
- **Sider**: 27 system organ classes where molecules are marketed drugs and adverse drug reactions [13].

Dataset Processing. The raw data of molecules are given as *SMILES* strings. We transfer *SMILES* strings to molecular graphs by using Rdkit.Chem [14]. Then we extracted a set of node and bond features which can preserve the molecular structure best to use in the experiments. The details about features are listed in Table 1.

5.2 Baselines

We compare our model with multiple baseline models.

¹<https://tripod.nih.gov/tox21/challenge/>

Table 2: The performances of all methods on both datasets. Our proposed method Meta-MGNN can outperform all baseline methods. The last column reports the average improvements (in percentage) of Meta-MGNN over the best baseline method in different tasks. Bond indicates the best performance. Underline represents the best baseline performance.

Dataset	Task	GraphSAGE [9] (2017)	GCN [12] (2017)	MAML [5] (2017)	Seq3seq [34] (2018)	EGNN [11] (2019)	PreGNN [10] (2020)	Meta-MGNN	Δ AUC
1-shot									
Tox21	SR-HS	65.97	65.00	68.56	<u>73.18</u>	72.51	73.09	73.81	+0.63
	SR-MMP	71.23	71.20	76.34	<u>79.08</u>	76.90	76.20	79.09	+0.01
	SR-p53	58.05	66.60	71.28	75.23	78.03	76.87	77.71	-0.32
	Average	65.10	67.60	72.06	<u>75.83</u>	75.81	75.39	76.87	+1.04
Sider	Si-T1	65.23	63.60	66.82	66.50	71.39	<u>73.04</u>	75.41	+2.37
	Si-T2	60.47	62.01	63.62	57.03	<u>67.87</u>	66.06	69.39	+1.52
	Si-T3	61.45	64.52	67.50	61.38	<u>68.23</u>	<u>70.36</u>	70.65	+0.29
	Si-T4	64.41	65.28	69.02	63.45	<u>72.67</u>	72.34	72.69	+0.02
	Si-T5	77.85	74.95	77.07	74.83	<u>78.88</u>	77.99	79.95	+1.07
	Si-T6	61.19	63.20	67.01	63.70	66.31	<u>69.45</u>	71.97	+2.52
	Average	65.10	65.60	68.51	64.48	70.89	<u>71.54</u>	73.34	+1.80
5-shots									
Tox21	SR-HS	69.09	68.13	69.02	<u>74.07</u>	73.23	73.39	74.80	+0.73
	SR-MMP	72.22	69.06	76.43	80.40	79.07	78.25	80.26	-0.14
	SR-p53	61.45	72.01	73.95	77.07	<u>78.12</u>	78.01	79.00	+0.88
	Average	67.59	69.73	73.13	<u>77.18</u>	76.81	76.55	78.02	+0.84
Sider	Si-T1	67.61	65.66	70.12	68.99	72.76	<u>74.77</u>	76.32	+1.55
	Si-T2	59.86	64.62	64.46	56.53	<u>68.13</u>	65.69	69.34	+1.21
	Si-T3	60.61	64.90	68.20	64.20	70.11	<u>71.07</u>	72.29	+1.22
	Si-T4	64.82	64.85	67.75	67.15	72.73	<u>73.42</u>	74.46	+1.04
	Si-T5	78.33	76.93	78.61	78.55	79.61	<u>80.67</u>	81.79	+1.12
	Si-T6	61.91	62.06	67.74	66.30	67.17	<u>71.48</u>	74.12	+2.64
	Average	65.52	66.50	69.48	66.95	71.75	<u>72.85</u>	74.72	+1.87

- **GraphSAGE [9]**. It generates the nodes' embedding by sampling and aggregating their neighbors' embeddings, which can effectively capture the graph information.
- **GCN [12]**. It is a widely used graph-based model, which contains an effective convolutional neural network component. GCN outperforms various models by learning both local graph structure and features of nodes.
- **MAML [5]**. It builds a task-agnostic algorithm for few-shot learning, where training a model's parameters using a small number of gradient updates will lead to fast learning on new tasks.
- **Seq3seq [43]**. It is a Seq2Seq model for molecular property prediction. The loss function contains both self-recovery loss and inference task loss.
- **EGNN [11]**. It is an edge-labeling graph neural network for few-shot learning which is proved a well-generalizable model for low-data problem.
- **PreGNN [10]**. This model develops self-supervised learning to pretrain GNN for molecular property prediction. It captures both useful local and global information.

5.3 Evaluation Metrics

We evaluate the performance of each model using ROC-AUC. We consider each molecular property as an independent task for few-shot learning. We use 3 and 6 tasks as test tasks of Tox21 and Sider data, respectively. Each task is a binary label classification task. Table 2 and Figure 3 report the result for each test task. For both datasets, we consider 2-way classification with 1 and 5 shots.

5.4 Reproducibility Settings

We take graph isomorphism network (GIN) [34] as base graph neural network. In our experiment, we utilize the supervised-contextpred pre-trained GIN of PreGNN [10]. The GIN layer number is set as 5. We set all embedding dimensions to 300. The same feature will share the same initial embedding. We set the update step in training tasks as 5 and the update step in testing tasks as 10. We set the trade-off weight of self-supervised module as 0.1. We use Pytorch to implement the model and run it on a GPU.

5.5 Comparisons with Baselines

Overall Performance. The overall performances of all methods are reported in Table 2. According to this table, we can find that Meta-MGNN outperforms all baseline models on both Tox21 and

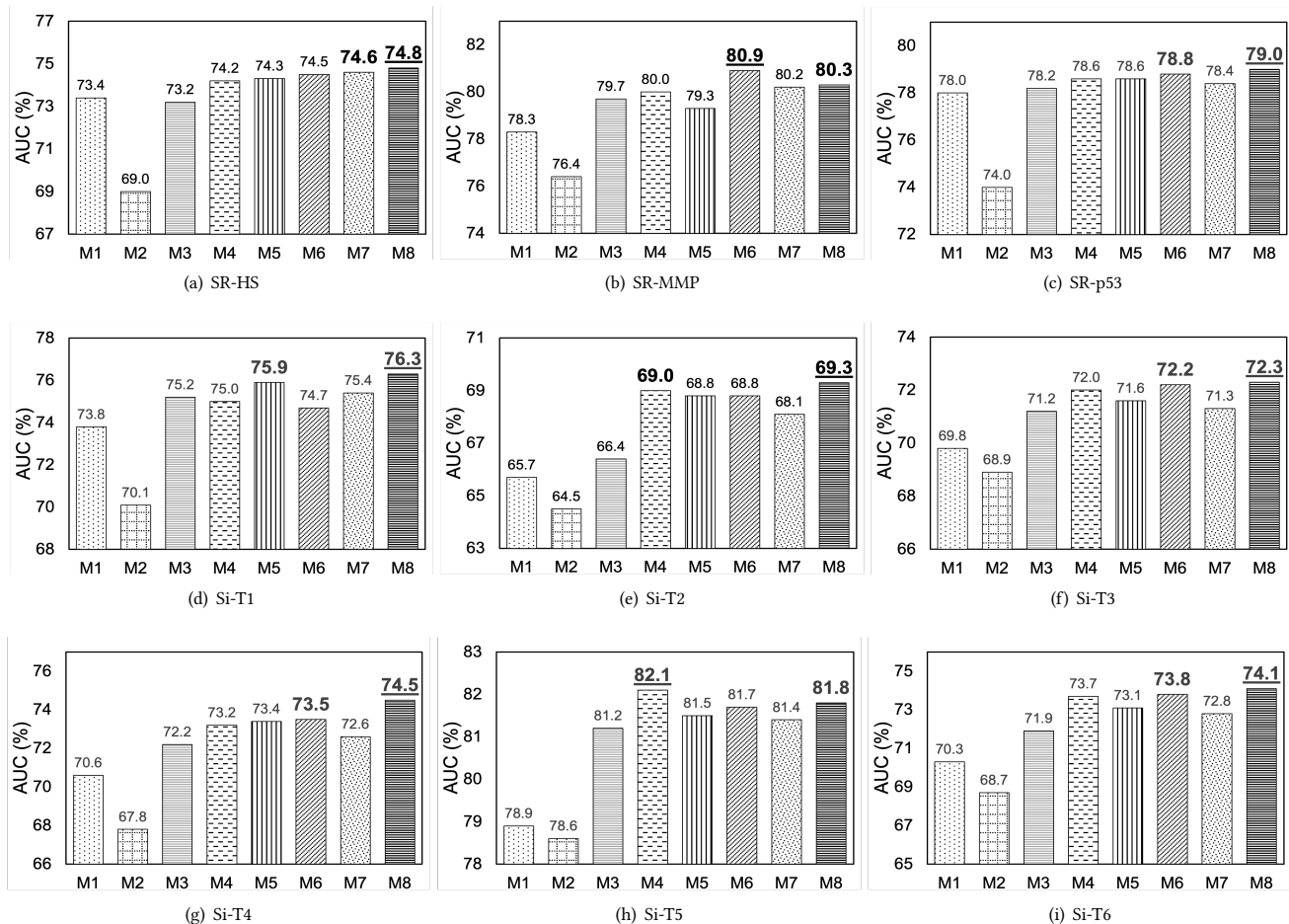


Figure 2: Performances of different model variants in Tox21 data (the first row) and Sider (the remaining figures) data. Different model components (i.e., graph neural network pre-training, self-supervised module, and task-aware attention) indeed make effect and improve the model performance. Our proposed model (M8) has better result than all other model variants.

Sider datasets. Specifically, for 1-shot learning, the average improvements are **+1.04%** and **+1.80%** on Tox21 and on Sider, respectively. The values equal **+0.84%** and **+1.87%** for 5-shot learning. In addition, we observe that PreGNN [10] and EGNN [11] perform the best among all baseline methods on average. However, the baseline methods do not have stable performance on different tasks. In other words, they may perform well on one task, but perform poorly on another task. In comparison, the performance of Meta-MGNN is stable. It has the best performance for all tasks in both dataset.

Analyzing Meta-MGNN Structure. MAML demonstrates superior performance than the other two GNN models (GraphSage and GCN). It makes sense since MAML trains the model through meta-learning, making it better adapt to new tasks with few data samples. Besides taking advantage of meta-learning, Meta-MGNN also utilizes pre-trained graph neural network model (PreGNN) [10] to initialize model parameters. PreGNN uses a large amount of molecules data to pre-train the graph neural network, which leads to better parameter initialization. Therefore, by taking advantages of both meta-learning and pre-training, Meta-MGNN demonstrates superior performance than the other baseline methods.

Table 3: We implemented 8 model variants for ablation study. The abbreviations used in the table: pre-trained model (PTM), meta-learning (ML), bond reconstruction (BR), atom-type prediction (AP) and task-aware attention (T-At).

	PTM	ML	BR	AP	T-At
M1	✓				
M2		✓			
M3	✓	✓			
M4	✓	✓	✓		
M5	✓	✓		✓	
M6	✓	✓	✓	✓	
M7	✓	✓			✓
M8	✓	✓	✓	✓	✓

Performance on Different Datasets. From Table 2, we can observe that our proposed Meta-MGNN outperforms the best baseline method by **+1.80%** for 1-shot learning and **+1.87%** for 5-shots learning on Sider dataset. However, the average improvements on Tox21 are **+1.04%** for 1-shot learning and **+0.84%** for 5-shots learning,

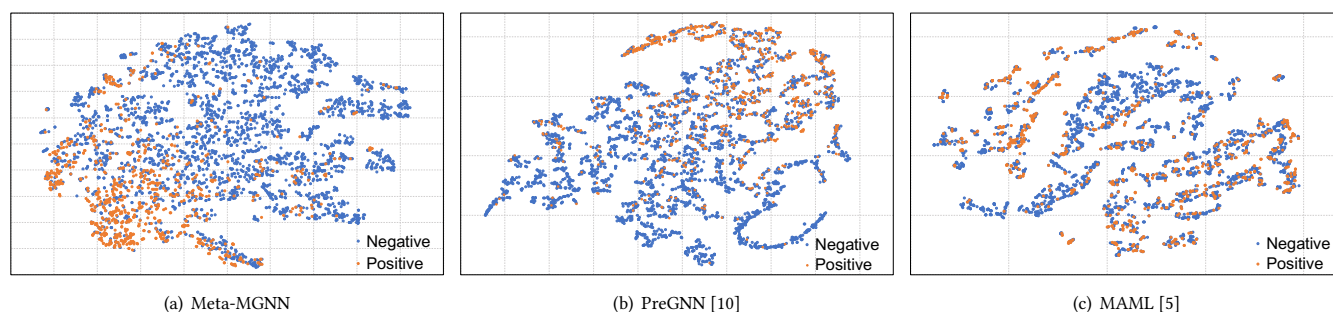


Figure 3: Visualizations of molecular embeddings generated by our model (Meta-MGNN), PreGNN [10], and MAML [5]. The blue dots denote negative labels in SR-MMP (a molecular property). The orange dots represent positive labels in SR-MMP. Our model can better discriminate embeddings of these two kinds of labels than the other methods.

which are smaller than those in Sider. As we know, the major advantage of few-shot learning model is to make model predict new tasks better by using a small number of data samples. Obviously, training with more tasks allows the model to learn more knowledge. The advantages of the few-shot learning model can be better reflected on the dataset which contains more tasks. Since Sider has more tasks than Tox21, Meta-MGNN can deliver greater improvements on Sider than on Tox21. Additionally, we also find that the overall performance on Tox21 is better than that on Sider for all models. This is due to the larger size of Tox21, which improves the generalization capabilities of these deep learning models, as reflected by the evaluation scores.

5.6 Ablation Study

Settings. Besides comparing with baseline methods, we also implement model variants (ablation studies) to show the effectiveness of different model components. The details of different model variants are illustrated as follows (also shown in the Table 3):

- **M1.** Pre-trained graph neural network. We take GIN [34] as our base graph neural network and pre-train it by both supervised and unsupervised (Context Prediction) pre-training strategies.
- **M2.** Graph neural network model (without pre-training) trained with the meta learning process.
- **M3.** Our base model, which is based on GIN [34] and learned with meta-learning algorithms. This model is also pre-trained by supervised and unsupervised (Context Prediction) pre-training.
- **M4 & M5 & M6.** These models are based on M3 and augmented with the self-supervised module. M4, M5, and M6 are augmented with the bond reconstruction, the atom-type prediction, and both of them, respectively. This is to analyze the effectiveness of the self-supervised module.
- **M7.** It is based on M3 and enhanced with task-aware attention to incorporate the importance of different tasks. This is to analyze the effectiveness of task weight in meta-learning.
- **M8.** It is based on M3 and augmented with both both self-supervised module and self-attentive task weight.

Performance Comparison and Analysis. The performances of all model variants are shown in Figure 2. The three sub-figures in the first row are model performance on Tox21. The sub-figures in the second row and third row are model performance on Sider. There are several findings from these figures. First, M2 has the worst

results in all cases, illustrating the significant impact of the pre-training step for graph neural networks. Second, the performance of M3 is better than M1 and M2, which indicates the effectiveness of combining both the pre-training and few-shot learning strategies. Third, adding different self-supervised components (bond reconstruction and atom type prediction) can further improve model performance, as reflected by the better performances of M4, M5, and M6 over M3. Among these three variants, M6 has the best performance as it adds both self-supervised tasks. Additionally, M7 outperforms M3, demonstrating the benefit of incorporating task-attention weight into the meta-learning process. At last, M8 (the proposed model) has the best performances in most cases, which shows the best capability graph neural network model trained by meta-learning process and augmented with both self-supervised module and task-aware attention. According to these findings, we can conclude that different model components indeed bring benefits to model design and improve performance.

5.7 Case Study of Embedding Visualization

To better show the effectiveness of our model, we visualize the molecular embeddings generated by our proposed Meta-MGNN, PreGNN [10] and MAML [5] using t-SNE [19], which are shown in Figure 3. Specifically, it shows the embedding result of testing datasets from SR-MMP (a molecular property). The blue plots and orange plots represent molecules without SR-MMP property and with SR-MMP property, respectively. It can be observed that our model achieves better performance in discriminating two kinds of molecules than the other two models. In Figure 3(a), the bottom left corner of the figure is mostly occupied by orange dots and the blues ones are mostly in the upper right corner. However, most orange plots are mixed with blue plots in Figure 3(b) and 3(c).

6 CONCLUSIONS

In this work, we proposed a few-shot learning approach for the molecular property prediction problem, which is important and has not been well studied. We proposed a novel model called Meta-MGNN. Meta-MGNN utilized a graph neural network (with pre-training) to learn molecular embeddings and further employed a meta-learning process to learn well-initialized model parameters that could be fast adapted to new molecular properties with few-shot data samples. A self-supervised module and self-attentive

task weight were further proposed and incorporate into the meta-learning framework, which benefited the whole model. We evaluated our model on two public multi-task datasets and the comparison of the experimental results showed that our model can outperform state-of-art methods. The effectiveness of each model component was also verified. The initial success of this study suggests following studies. The future work might consider better task embedding formulation when computing task weight in meta-learning. It is also possible to fuse both graph and sequence model to learn molecular embeddings for the meta-learning process.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grant CCI-1925607. We thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

- [1] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. 2017. Low data drug discovery with one-shot learning. In *ACS Central Science*.
- [2] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. 2018. Meta-learning with differentiable closed-form solvers. In *International Conference for Learning Representation (ICLR)*.
- [3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.
- [4] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The Web Conference*.
- [5] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*.
- [6] Victor Garcia and Joan Bruna. 2018. Few-shot learning with graph neural networks. In *International Conference for Learning Representation (ICLR)*.
- [7] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*.
- [8] Zhichun Guo, Wenhao Yu, Chuxu Zhang, Meng Jiang, and Nitesh Chawla. 2020. GraSeq: graph and sequence fusion learning for molecular property prediction. In *International Conference on Information and Knowledge Management (CIKM)*.
- [9] Will Hamilton, Zitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*.
- [10] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. In *International Conference for Learning Representation (ICLR)*.
- [11] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. 2019. Edge-labeling graph neural network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [12] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference for Learning Representation*.
- [13] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. 2016. The SIDER database of drugs and side effects. In *Nucleic Acids Research*.
- [14] Greg Landrum. 2013. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling.
- [15] Yoonho Lee and Seungjin Choi. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*.
- [16] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference for Learning Representation (ICLR)*.
- [17] Ke Liu, Xiangyan Sun, Lei Jia, Jun Ma, Haoming Xing, Junqiu Wu, Hua Gao, Yax Sun, Florian Boulois, and Jie Fan. 2019. Chemo-Net: a molecular graph convolutional network for accurate drug property prediction. In *International Journal of Molecular Sciences*.
- [18] Chengqiang Lu, Qi Liu, Chao Wang, Zhenya Huang, Peize Lin, and Lixin He. 2019. Molecular property prediction: A multilevel quantum interactions modeling perspective. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- [19] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. In *Journal of Machine Learning Research*.
- [20] Elman Mansimov, Omar Mahmood, Seokho Kang, and Kyunghyun Cho. 2019. Molecular geometry prediction using a deep generative graph neural network. In *Scientific Reports*.
- [21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [22] Sereina Riniker and Gregory A Landrum. 2013. Similarity maps—a visualization strategy for molecular fingerprints and machine-learning methods. In *Journal of Cheminformatics*.
- [23] Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. 2014. Computational methods in drug discovery. In *Pharmacological Reviews*.
- [24] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *The ACM International Conference on Web Search and Data Mining*.
- [25] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [26] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. 2019. Applications of machine learning in drug discovery and development. In *Nature Reviews Drug Discovery*.
- [27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference for Learning Representation (ICLR)*.
- [28] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [29] Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. 2019. SMILES-BERT: Large scale unsupervised pre-training for molecular property prediction. In *ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB)*.
- [30] Michael J Waring, John Arrowsmith, Andrew R Leach, Paul D Leeson, Sam Mandrell, Robert M Owen, Garry Pairaudeau, William D Pennie, Stephen D Pickett, Jibo Wang, et al. 2015. An analysis of the attrition of drug candidates from four major pharmaceutical companies. In *Nature Reviews Drug Discovery*.
- [31] David Weininger, Arthur Weininger, and Joseph L Weininger. 1989. SMILES. 2. Algorithm for generation of unique SMILES notation. In *Journal of Chemical Information and Computer Sciences*.
- [32] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. In *IEEE Transactions on Neural Networks and Learning Systems*.
- [33] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. In *Chemical Science*.
- [34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *International Conference for Learning Representation (ICLR)*.
- [35] Zheng Xu, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2017. Seq2seq fingerprint: An unsupervised deep molecular embedding for drug discovery. In *ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB)*.
- [36] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The Web Conference (WWW)*.
- [37] Huaxiu Yao, Chuxu Zhang, Ying Wei, Meng Jiang, Suhang Wang, Junzhou Huang, Nitesh Chawla, and Zhenhui Li. 2020. Graph few-shot learning via knowledge transfer. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- [38] Wenhao Yu, Mengxia Yu, Tong Zhao, and Meng Jiang. 2020. Identifying referential intention with heterogeneous contexts. In *The Web Conference (WWW)*.
- [39] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.
- [40] Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. 2020. Few-shot knowledge graph completion. In *AAAI Conference on Artificial Intelligence (AAAI)*.
- [41] Chuxu Zhang, Lu Yu, Mandana Saebi, Meng Jiang, and Nitesh Chawla. 2020. Few-shot multi-hop relation reasoning over knowledge bases. In *Conference on Empirical Methods in Natural Language Processing: Findings (EMNLP: Findings)*.
- [42] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. 2018. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [43] Xiaoyu Zhang, Sheng Wang, Feiyun Zhu, Zheng Xu, Yuhong Wang, and Junzhou Huang. 2018. Seq3seq fingerprint: towards end-to-end semi-supervised deep drug discovery. In *ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics (BCB)*.
- [44] Tong Zhao, Bo Ni, Wenhao Yu, and Meng Jiang. 2020. Early anomaly detection by learning and forecasting behavior. In *arXiv preprint arXiv:2010.10016*.
- [45] Shuangjia Zheng, Xin Yan, Yuedong Yang, and Jun Xu. 2019. Identifying structure-property relationships through SMILES syntax analysis with self-attention mechanism. In *Journal of Chemical Information and Modeling*.
- [46] Fan Zhou, Chengtai Cao, Kunpeng Zhang, Goce Trajcevski, Ting Zhong, and Ji Geng. 2019. Meta-GNN: On few-shot node classification in graph meta-learning. In *International Conference on Information and Knowledge Management (CIKM)*.