KPGT: Knowledge-Guided Pre-training of Graph Transformer for Molecular Property Prediction

Han Li
lihan19@mails.tsinghua.edu.cn
Institute for Interdisciplinary Information Sciences, Tsinghua University
Haidian Qu, Beijing Shi, China

Dan Zhao*
zhaodan2018@tsinghua.edu.cn
Institute for Interdisciplinary Information Sciences, Tsinghua University
Haidian Qu, Beijing Shi, China

Jianyang Zeng*
zengjy321@tsinghua.edu.cn
Institute for Interdisciplinary Information Sciences, Tsinghua University
Haidian Qu, Beijing Shi, China

ABSTRACT

Designing accurate deep learning models for molecular property prediction plays an increasingly essential role in drug and material discovery. Recently, due to the scarcity of labeled molecules, self-supervised learning methods for learning generalizable and transferable representations of molecular graphs have attracted lots of attention. In this paper, we argue that there exist two major issues hindering current self-supervised learning methods from obtaining desired performance on molecular property prediction, that is, the ill-defined pre-training tasks and the limited model capacity. To this end, we introduce Knowledge-guided Pre-training of Graph Transformer (KPGT), a novel self-supervised learning framework for molecular graph representation learning, to alleviate the aforementioned issues and improve the performance on the downstream molecular property prediction tasks. More specifically, we first introduce a high-capacity model, named Line Graph Transformer (LiGhT), which emphasizes the importance of chemical bonds and is mainly designed to model the structural information of molecular graphs. Then, a knowledge-guided pre-training strategy is proposed to exploit the additional knowledge of molecules to guide the model to capture the abundant structural and semantic information from large-scale unlabeled molecular graphs. Extensive computational tests demonstrated that KPGT can offer superior performance over current state-of-the-art methods on several molecular property prediction tasks.

CCS CONCEPTS

• **Computing methodologies** → **Unsupervised learning**; **Learning latent representations**; **Neural networks**.

KEYWORDS

graph representation learning, self-supervised learning, graph transformer, graph pre-training, molecular property prediction

ACM Reference Format:
Han Li, Dan Zhao, and Jianyang Zeng. 2022. KPGT: Knowledge-Guided Pre-training of Graph Transformer for Molecular Property Prediction. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539426

*Dan Zhao and Jianyang Zeng are the corresponding authors.

1 INTRODUCTION

Molecular property prediction is of great significance to design novel molecules with desired properties for drug and material discovery [48, 55]. Representing molecules as graphs, where nodes correspond to the atoms and edges correspond to the chemical bonds, deep learning models, especially graph neural networks (GNNs), have been widely used for molecular graph representation learning [4, 8, 17, 56, 60, 64]. However, due to the limited amount of labeled data and the giant chemical space, such deep learning models trained with supervised learning strategies often perform poorly, especially on the prediction of out-of-distribution data samples [25, 55].

Following the significant success of self-supervised learning methods in the fields of natural language processing (NLP) [10, 13] and computer vision (CV) [22, 23], many recent works have proposed to employ self-supervised learning strategies to pre-train GNNs by leveraging the large-scale unlabeled molecules and have achieved superior prediction performance in comparison with supervised learning methods on the downstream molecular property prediction tasks [25, 31, 32, 44, 46, 53, 58, 62, 63]. Despite such fruitful progress, the prediction performance is still far from ideal. In this paper, we argue that current self-supervised learning methods on molecular graphs still encounter the following two main issues:

The ill-defined pre-training tasks. The performance of self-supervised learning methods crucially depends on the design of pre-training tasks. So far, the self-supervised learning methods on molecular graphs can be roughly divided into two categories, that is, generative and contrastive methods according to their design of pre-training tasks. The generative methods follow the masked-language models in the NLP field, e.g., BERT [10], through masking a portion of molecular graphs, e.g., edges, nodes or subgraphs, and then learning to retrieve the original graphs [25, 44]. On the other hand, following the pioneering works in CV, the contrastive methods on molecular graphs first generate graph augmentations through strategies like node replacing, node dropping and edge perturbation for molecular graphs, and then learn to match the augmented graphs with the corresponding original molecular graphs in the embedding space [62, 63]. However, unlike the word masking of human languages and the augmentation of images (e.g., resizing and rotation), which do not impact the fundamental semantics of the raw inputs, a small modification of molecular graphs can greatly change the characteristics of the corresponding molecules (an illustrative example is shown in Figure 4). Therefore, current self-supervised learning methods on molecular graphs can only capture the structural similarity of graphs and the simple construction rules (e.g., valency rule) of molecules, but fail to induce the abundant semantics related to molecular properties from chemical structures which are potentially more important to the downstream learning tasks.

The limited model capacity. Due to the giant chemical space and the wide breadth of the molecular properties ranging from quantum mechanics, physical chemistry, biophysics to physiology [55], a high-capacity model is required to capture sufficient information from enormous unlabeled molecules. Meanwhile, the success of self-supervised learning methods in the NLP and CV domains is indispensable to the emerging backbone networks with increasing amounts of parameters [12, 33]. Especially, transformer-based models have been proven to predominantly yield excellent prediction performance in these fields. However, applying the transformer-based architectures in the self-supervised learning on molecular graphs is underexplored. Most of the previous defined self-supervised learning methods on molecular graphs mainly employ GNNs, e.g., Graph Isomorphism Network (GIN) [57], as the backbone networks, which provide only limited model capacity and potentially fail to capture the wide range of the information required for the prediction of various properties for a diversity of molecules.

To this end, we propose Knowledge-guided Pre-training of Graph Transformer (KPGT), a novel self-supervised learning framework, to alleviate the aforementioned issues and learn more generalizable, transferable and robust representations of molecular graphs to improve the performance of the downstream molecular property prediction tasks. First, we propose a high-capacity model, named Line Graph Transformer (LiGhT), which represents molecular graphs as line graphs to emphasize the importance of chemical bonds, and also introduces path encoding and distance encoding to accurately preserve the structural information of molecules. Then, we design a knowledge-guided pre-training strategy based on a generative self-supervised learning scheme. Instead of directly predicting the randomly masked nodes using the contextual information, our proposed strategy leverages the additional knowledge of molecules (i.e., molecular descriptors and fingerprints), which serves as the semantics lost in the masked graph to guide the prediction of the masked nodes, thus making the model capture the abundant structural and semantic information from large-scale unlabeled molecules.

We conducted extensive computational experiments to evaluate the performance of our proposed method. We first pre-trained KPGT on a large-scale unlabeled molecule dataset and then applied the pre-trained model to eleven downstream molecular property datasets. The test results demonstrated that KPGT achieved superior results in comparison with current state-of-the-art self-supervised learning methods and thus verified the effective design of KPGT.

To summarize, our work makes the following three main contributions: (1) We point out that current self-supervised learning methods on molecular graphs have two major issues, that is, the ill-defined pre-training tasks and the limited model capacity; (2) we propose KPGT, a novel self-supervised learning framework, consisting of a novel graph transformer architecture, LiGhT, and a knowledge-guided pre-training strategy, to alleviate the current issues in the representation learning of molecular graphs and improve the performance on the downstream molecular property prediction tasks; (3) we conduct extensive tests to demonstrate that our proposed method can offer superior performance over current state-of-the-art methods on several molecular property prediction tasks. The source code and datasets to reproduce our computational test results are available at https://github.com/lihan97/KPGT.

2 RELATED WORK

2.1 Molecular Representation Learning

Many efforts have been devoted to improving molecular representation learning for accurate molecular property prediction. The early feature-based methods exploited fixed molecular representation, e.g., molecular descriptors and fingerprints, to represent molecules in a vector space [5, 11, 49]. This kind of method highly relied on complicated feature engineering to produce promising prediction performance. Then, deep learning based models were introduced to yield more expressive molecular representations. In particular, with their great advantage in modeling graph-structured data, GNNs had been widely applied for molecular graph representation learning. For example, Gilmer et al. [17] introduced a message passing framework for molecular property prediction. Then many works proposed to modify the message passing mechanism to improve the expressiveness of GNNs [4, 8, 56]. Please refer to [54] for a compact review on molecular property prediction with GNNs. The message passing operators employed in the GNNs aggregated only local information and failed to capture the long-range dependence in molecules. Recently, graph transformers have emerged to model such long-range dependencies. The applications of transformer-based architectures to molecular graphs have remained limited mainly due to the difficulty of properly modeling the structural information of molecular graphs. Therefore, many efforts have been devoted to accurately preserving the structural information of molecular graphs [7, 28, 35, 36, 61].

2.2 Self-Supervised Learning on Molecular Graphs

Due to the data scarcity and the giant chemical space, self-supervised learning on molecular graphs has emerged as a core direction recently. Current self-supervised learning methods on molecular graphs can be roughly categorized into generative and contrastive ones, where they differed in the design of the pre-training tasks. The generative methods followed the masked-language model in the NLP field through masking nodes, edges or subgraphs and learning to retrieve the original graphs [25, 44]. On the other hand, the contrastive methods followed the pioneer works in the CV field. For example, You et al. [63] designed a framework with four graph augmentation strategies, that is, node dropping, edge perturbation, attribute masking and subgraph generating. You et al. [62] further improved this framework by automatically and adaptively selecting data augmentations. In addition, Xu et al. [58] proposed to learn the hierarchical prototypes upon graph embeddings to infer the global-semantic structure in molecular graphs. Moreover, Stärk et al. [46] and Liu et al. [32] exploited the 3D geometry of molecules to improve the pre-training of GNNs.

3 KNOWLEDGE-GUIDED PRE-TRAINING OF GRAPH TRANSFORMER (KGPT)

In this section, we elaborate on the design of our self-supervised learning framework, Knowledge-guided Pre-training of Graph Transformer (KPGT). First, we introduce how we transform molecular graphs into line graphs. Then we provide the detailed implementation of our proposed Line Graph Transformer (LiGhT) for encoding the molecular line graphs. Finally, we introduce our knowledge-guided pre-training strategy.

3.1 Line Graph Transformer (LiGhT)

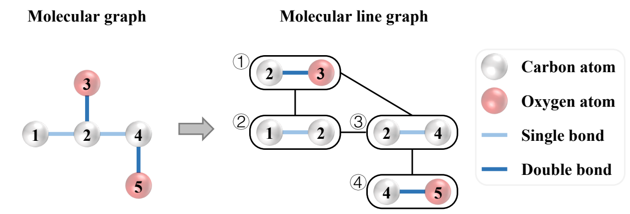

Figure 1: An illustrative example of the transformation of a molecular graph to a molecular line graph.

3.1.1 Molecular Line Graph. Given a molecule, it can be represented as a molecular graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i\}_{i\in[1,N_v]}$ stands for the set of nodes (i.e., atoms), $\mathcal{E} = \{e_{i,j}\}_{i,j\in[1,N_v]}$ stands for the set of edges (i.e., chemical bonds), and N_v stands for the number of nodes. The initial features of node v_i are represented by $\boldsymbol{x}_i^v \in \mathbb{R}^{D_v}$ and the initial features of edge $e_{i,j}$ are represented by $\boldsymbol{x}_{i,j}^e \in \mathbb{R}^{D_e}$, where D_v and D_e stand for the dimensions of features of nodes and edges, respectively.

As an illustrative example shown in Figure 1, a molecular graph $\mathcal{G}$ can be transformed to a molecular line graph $\hat{\mathcal{G}} = \{\hat{\mathcal{V}}, \hat{\mathcal{E}}\}$ as follows: (1) For each edge in $\mathcal{G}$, e.g., $e_{i,j}$, create a node $\hat{v}_{i,j}$ in $\hat{\mathcal{G}}$; (2) for every two edges in $\mathcal{G}$ that have a node in common, create an edge between their corresponding nodes in $\hat{\mathcal{G}}$.

Then, for each node $\hat{v}_{i,j}$ in $\hat{\mathcal{G}}$, its initial feature embedding $\boldsymbol{h}_{\hat{v}_{i,j}} \in \mathbb{R}^{D_{\hat{v}}}$ is defined as:

$$\boldsymbol{h}_{\hat{v}_{i,j}} = concat(\boldsymbol{W}_v \boldsymbol{x}_i^v + \boldsymbol{W}_v \boldsymbol{x}_j^v, \boldsymbol{W}_e \boldsymbol{x}_{i,j}^e), \tag{1}$$

where $D_{\hat{v}}$ stands for the dimension of the feature embeddings of nodes in molecular line graphs, $\boldsymbol{W}_v \in \mathbb{R}^{\frac{D_{\hat{v}}}{2}\times D_v}$ and $\boldsymbol{W}_e \in \mathbb{R}^{\frac{D_{\hat{v}}}{2}\times D_e}$ stand for the trainable projection matrices, and $concat(\cdot)$ stands for the concatenation operator. For clarity, we denote the nodes in a molecular line graph as $\hat{\mathcal{V}} = \{\hat{v}_i\}_{i\in[1,N_{\hat{v}}]}$ in the following sections, where $N_{\hat{v}}$ stands for the number of nodes in the molecular line graph.

3.1.2 Graph Transformer Architecture. In this section, we propose Line Graph Transformer (LiGhT), a novel transformer-based architecture, to encode the molecular line graphs and learn global feature representations for molecular property prediction (see Figure 2). LiGhT is built upon a classic transformer encoder initially proposed in [50]. The classic transformer consists of multiple transformer layers, each of which is composed of a multi-head self-attention

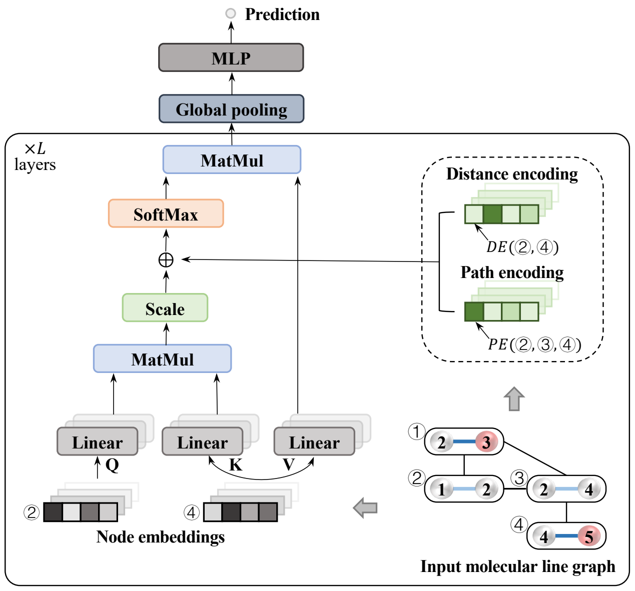

Figure 2: An illustration of the Line Graph Transformer (LiGhT) architecture. LiGhT is on the basis of a classic transformer encoder and introduces path encoding (PE) and distance encoding (DE) into the multi-head self-attention module to capture the structural information of molecules.

module, followed by a feed-forward network (FFN) that includes residual connection and layer normalization operations.

More specifically, given a molecular line graph and the corresponding node feature matrix $\boldsymbol{H} \in \mathbb{R}^{N_{\hat{v}}\times D_{\hat{v}}}$, the multi-head self-attention module at layer l is calculated as:

$$\begin{aligned} \boldsymbol{Q}^{l,k} &= \boldsymbol{H}^{l-1}\boldsymbol{W}_Q^{l,k}, \boldsymbol{K}^{l,k} = \boldsymbol{H}^{l-1}\boldsymbol{W}_K^{l,k}, \boldsymbol{V}^{l,k} = \boldsymbol{H}^{l-1}\boldsymbol{W}_V^{l,k}, \\ \boldsymbol{A}^{l,k} &= softmax(\frac{\boldsymbol{Q}^{l,k}(\boldsymbol{K}^{l,k})^T}{\sqrt{D_h}}), \boldsymbol{H}^{l,k} = \boldsymbol{A}^{l,k}\boldsymbol{V}^{l,k}, \\ \boldsymbol{H}^l &= concat(\boldsymbol{H}^{l,1}, \boldsymbol{H}^{l,2}, \ldots, \boldsymbol{H}^{l,N_h}), \end{aligned} \tag{2}$$

where $\boldsymbol{H}^{l-1}$ stands for the node feature matrix at the $(l-1)$-th layer, $\boldsymbol{W}_Q^{l,k} \in \mathbb{R}^{D_{\hat{v}}\times D_h}$, $\boldsymbol{W}_K^{l,k} \in \mathbb{R}^{D_{\hat{v}}\times D_h}$ and $\boldsymbol{W}_V^{l,k} \in \mathbb{R}^{D_{\hat{v}}\times D_h}$ stand for the trainable projection matrices of the k-th head at layer l, $D_h = \frac{D_{\hat{v}}}{N_h}$ stands for the dimension of each self-attention head, N_h stands for the number of self-attention heads, and $softmax(\cdot)$ stands for the softmax operator. The output $\boldsymbol{H}^l$ is then passed to a FFN:

$$\begin{aligned} \hat{\boldsymbol{H}}^l &= LN(\boldsymbol{H}^{l-1} + \boldsymbol{H}^l), \\ \boldsymbol{H}^l &= LN(\boldsymbol{W}_2^l GELU(\boldsymbol{W}_1^l \hat{\boldsymbol{H}}^l) + \hat{\boldsymbol{H}}^l), \end{aligned} \tag{3}$$

where $LN(\cdot)$ stands for the LayerNorm operator [3], $GELU(\cdot)$ stands for the GELU activation function [24], and $\boldsymbol{W}_1^l \in \mathbb{R}^{4D_{\hat{v}}\times D_{\hat{v}}}$ and $\boldsymbol{W}_2^l \in \mathbb{R}^{D_{\hat{v}}\times 4D_{\hat{v}}}$ stand for the trainable projection matrices at layer l.

Unlike GNNs, which aggregate information from neighboring nodes, the global receptive field of the transformer enables each node to attend to the information at any position in one transformer layer. Nevertheless, such a classic transformer architecture ignores the connectivity of graphs, thus causing a significant loss in the structural information of molecules. In the NLP field, the structural information of chain-structured language can be preserved by giving each token a position embedding or encoding the relative distance between any two tokens [45, 50], which actually cannot be easily generalized to the graph-structured data. Here, we introduce path encoding and distance encoding to inject the inductive bias in the multi-head self-attention module and thus enable LiGhT to encode the structural information of the molecular line graphs.

Path Encoding. For each pair of nodes $\hat{v}_i$ and $\hat{v}_j$ in the molecular line graph, we first derive the shortest path between them and then encode the path features to an attention scalar $a^p_{i,j}$ in a path attention matrix $\boldsymbol{A}^p \in \mathbb{R}^{N_{\hat{v}} \times N_{\hat{v}}}$ as follows:

$$\begin{aligned}(\hat{v}^p_1, \hat{v}^p_2, \ldots, \hat{v}^p_{N_p}) &= SP(\hat{v}_i, \hat{v}_j), \\ a^p_{i,j} &= \boldsymbol{W}^p_a \frac{1}{N_p} \sum_{n=1}^{N_p} \boldsymbol{W}^p_n \boldsymbol{h}_{v^p_n},\end{aligned} \tag{4}$$

where $SP(\cdot)$ stands for the shortest path function implemented by networkx [19], $(\hat{v}^p_1, \hat{v}^p_2, \ldots, \hat{v}^p_{N_p})$ stands for the shortest path between $\hat{v}_i$ and $\hat{v}_j$, N_p stands for the length of the path, $\boldsymbol{h}_{v^p_n}$ stands for the feature of the n-th node in the shortest path, $\boldsymbol{W}^p_n \in \mathbb{R}^{D_p \times D_{\hat{v}}}$ stands for the trainable projection matrix for the n-th node in the path, $\boldsymbol{W}^p_a \in \mathbb{R}^{1 \times D_p}$ stands for a trainable projection matrix to project the path embedding to an attention scalar, and D_p stands for the dimension of the path embedding.

Distance Encoding. Following [35, 61], we also leverage the distances between pairs of nodes to further encode the spatial relations in the molecular line graphs. More specifically, given nodes $\hat{v}_i$ and $\hat{v}_j$ in a molecular line graph, we encode their distance to an attention scalar $a^d_{i,j}$ in a distance attention matrix $\boldsymbol{A}^d \in \mathbb{R}^{N_{\hat{v}} \times N_{\hat{v}}}$ as follows:

$$\begin{aligned}d_{i,j} &= SPD(\hat{v}_i, \hat{v}_j), \\ a^d_{i,j} &= \boldsymbol{W}^d_2 GELU(\boldsymbol{W}^d_1 d_{i,j}),\end{aligned} \tag{5}$$

where $SPD(\cdot)$ stands for the shortest path distance functoin, $d_{i,j}$ stands for the derived distance between $\hat{v}_i$ and $\hat{v}_j$, $\boldsymbol{W}^d_1 \in \mathbb{R}^{D_d \times 1}$ and $\boldsymbol{W}^d_2 \in \mathbb{R}^{1 \times D_d}$ stand for the trainable projection matrices, and D_d stands for the dimension of the distance embedding.

Then, to introduce the encoded structural information into the model, we rewrite the formula of the attention matrix $\boldsymbol{A}^{l,k} \in \mathbb{R}^{N_{\hat{v}} \times N_{\hat{v}}}$ in the Eq. 2 as follows:

$$\boldsymbol{A}^{l,k} = softmax\left(\frac{\boldsymbol{Q}^{l,k}(\boldsymbol{K}^{l,k})^T}{\sqrt{D_h}} + \boldsymbol{A}^p + \boldsymbol{A}^d\right). \tag{6}$$

Here, we discuss the main advantages of our proposed model compared with the previously defined graph transformers:

First, by representing molecular graphs as line graphs, LiGhT emphasizes the importance of chemical bonds in molecules. Chemical bonds are the lasting attractions between atoms, which can be categorized into various types according to the ways they hold atoms together resulting in different properties of the formed molecules. However, the previously defined transformer architectures either omit the edge features or only introduce chemical bonds as the bias in the self-attention module, ignoring the rich information from chemical bonds [7, 28, 35, 36, 61]. In our case, LiGhT fills this gap and fully exploits the intrinsic features of chemical bonds.

Second, although strategies like path encoding have already been proposed in previous graph transformer architectures [7, 61], when encoding the paths, they only consider the edge features and ignore the node features in the paths. On the other hand, our path encoding strategy incorporates the features of the complete paths between pairs of nodes, thus encoding the structural information more precisely compared to the previous methods.

In summary, LiGhT provides a reliable backbone network for accurately modeling the structural and semantic information of molecular line graphs.

3.2 Knowledge-Guided Pre-training Strategy

Our pre-training strategy is derived mainly based on a generative self-supervised learning scheme. More specifically, we randomly choose a proportion of nodes in molecular line graphs for prediction. Following [10], for a chosen node, we first replace it with (1) a mask token 80% of the time, (2) a random node 10% of the time, and (3) the unchanged node 10% of the time, and then predict the type of the original node with a cross-entropy loss.

As mentioned above, directly applying such masking operations may lose the initial semantics of molecular graphs, thus making current generative methods ill-defined. In addition, without semantics, the only dependence between the masked nodes and their adjacent nodes is the valency rule, which is much weaker in comparison with the dependence between neighboring tokens in natural languages or neighboring pixels in images. As an example shown in Figure 5, if we mask an oxygen atom in the molecular graph, the oxygen atom is the expected prediction but many other atoms can also construct valid molecules according to the valency rules. Therefore, such weak dependence cannot always guide the model to make the right predictions, which may cause the model to simply memorize the whole dataset.

To alleviate these issues, we introduce additional knowledge of molecules in our pre-training framework, which serves as the semantics lost in the masked graphs to guide the prediction of the masked nodes.

For molecules, there is a large number of well-defined molecular descriptors and fingerprints that have been widely used to represent molecular characteristics and have been proven to be effective for molecular property prediction in the previous research [5, 11, 49]. Molecular descriptors are typically generated by logical and mathematical procedures to quantitatively describe the physical and chemical profiles of molecules [59]. For example, LogP is a type of molecular descriptor measuring the lipophilicity of the molecules. The molecular fingerprints describe a molecular structure by a bit string, where one indicates the existence of one substructure and vice versa [6, 43]. Therefore, molecular descriptors and fingerprints can provide valuable global and local information of molecules. Moreover, most of the molecular descriptors and fingerprints are readily available by current cheminformatics tools [18, 38, 41]. Thus, it will not introduce extra budget in the pre-training process.

To incorporate such additional knowledge in our pre-training framework, we define a special node for each molecular line graph, named knowledge node (K node), and make connections between

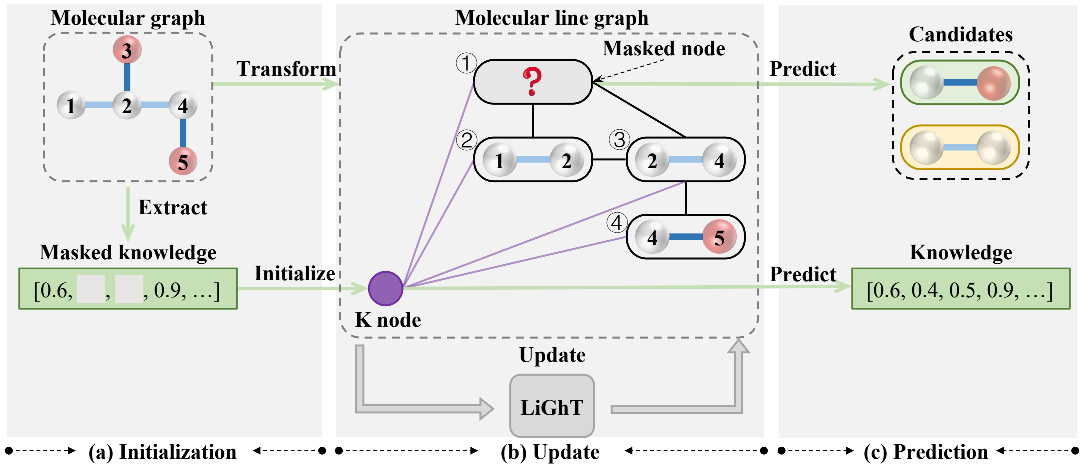

Figure 3: An illustration of the knowledge-guided pre-training strategy. For a molecular graph, we first transform it to a line graph and extract its additional knowledge. Then, we randomly mask a proportion of the nodes in the line graph and the same proportion of the additional knowledge and utilize the masked knowledge to initialize the virtually constructed K node. Next, LiGhT is employed to update the feature embeddings of nodes in the graph. Finally, the derived node embeddings of the masked nodes and K node are used to predict the node types and masked knowledge, respectively.

K node and each node individually. The raw features of the K node are initialized by the quantitative molecular descriptors and fingerprints. As such, in the pre-training, the extra knowledge can be attended by other nodes through the self-attention module in the transformer and thus guide the prediction of the masked nodes.

In the pre-training, we also randomly mask a proportion of the initial features of K nodes and learn to predict the masked molecular descriptors and fingerprints. The prediction of the masked molecular descriptors is formulated as a regression task equipped with an RMSE loss, while the prediction of fingerprints is formulated as a binary classification task equipped with a cross-entropy loss.

An illustrative example of our self-supervised learning strategy is shown in Figure 3. In summary, by explicitly introducing additional knowledge in the pre-training, KPGT is able to correct the current ill-defined generative pre-training tasks and thus enables the model to capture both the abundant structural and the rich semantics information from the molecular graphs, which can be essential to the downstream molecular property prediction tasks.

4 EXPERIMENTS

4.1 Experimental Settings

Dataset. For pre-training, we used two million unique unlabeled molecular SMILES samples from the ChEMBL29 dataset [14]. We employed RDKit [18] to abstract the molecular graphs from the SMILES strings and initialize the features of nodes and edges (see Appendix A.2.1 for more details).

After pre-training, we evaluated the effectiveness of KPGT on eleven molecular property datasets that have been widely used in the previous research, including eight classification datasets and three regression datasets, covering properties from multiple domains (see Appendix A.3 for more details about these datasets).

In practice, the test molecules can be structurally different from the training molecules. Therefore, we used scaffold splitting, which split a dataset according to the graph substructures of the molecules in the dataset, offering a more challenging yet realistic scenario. We split each dataset into training, validation and test sets with a ratio of 8:1:1. Following the previous research [44, 55], for each dataset, we performed three repeated tests with three randomly seeded splittings and then reported the means and standard deviations.

Additional Knowledge. In this work, we did not carefully collect the additional knowledge, though it would better benefit KPGT, intuitively. More specifically, we collected two sets of knowledge derived from RDKit [18]. The first set contained 200 molecular descriptors that had been widely used for molecular property prediction [44, 60]. The second set contained the most common 512 RDKit fingerprints, where each bit indicated the existence of a specific path of length between one and seven in the molecules.

Training Details. We mainly used Pytorch [39] and DGL [52] to implement KPGT. More specifically, we implemented a 12-layer LiGhT as the backbone network with a hidden size of 768 and the number of self-attention heads as 12. A mean pooling operation that averaged all the nodes in individual graphs was applied on top of the model to extract molecular representations. An Adam optimizer [26] with weight decay $1e^{-6}$ and learning rate $2e^{-4}$ was used to optimize the model. The model was trained with batch size 1024 for a total of 100,000 steps. The KPGT had around 100 million parameters. We set the masking rate of both nodes and additional knowledge to 0.5. The pre-training of KPGT took about two days on four Nvidia A100 GPUs.

For the downstream fine-tuning tasks, we added a predictor, which was a 2-layer multi-layer perceptron (MLP) with ReLU activation [2], on top of the base feature extractors. The cross-entropy loss and RMSE loss were implemented for binary classification tasks and regression tasks, respectively. The model was trained using an Adam optimizer with batch size 32 for another 50 epochs. Early stopping was employed with patience set to 20. For each method, we

Table 1: The AUROC performance of different methods on the classification datasets under both feature extraction and transfer learning settings. The higher result is better (marked by ↑). AVG represents the averaging results over all the datasets. The numbers in brackets are the standard deviations. The best result for each dataset is marked in bold and the second-best result is underlined.

Method	Classification dataset ↑								
	Feature extraction setting								
	BACE	BBBP	ClinTox	SIDER	Estrogen	MetStab	Tox21	ToxCast	**AVG**
Infomax [51]	$0.825_{(0.033)}$	$0.833_{(0.017)}$	$0.637_{(0.027)}$	$0.573_{(0.016)}$	$0.878_{(0.061)}$	$0.763_{(0.044)}$	$0.764_{(0.038)}$	$0.662_{(0.010)}$	0.742
Edgepred [20]	$0.807_{(0.014)}$	$0.823_{(0.011)}$	$0.637_{(0.059)}$	$0.545_{(0.019)}$	$0.851_{(0.070)}$	$0.792_{(0.064)}$	$0.702_{(0.040)}$	$0.620_{(0.004)}$	0.722
Masking [25]	$0.739_{(0.059)}$	$0.823_{(0.031)}$	$0.564_{(0.089)}$	$0.555_{(0.032)}$	$0.805_{(0.075)}$	$0.588_{(0.102)}$	$0.722_{(0.017)}$	$0.591_{(0.023)}$	0.673
Contextpred [25]	$0.822_{(0.053)}$	$0.875_{(0.009)}$	$0.607_{(0.062)}$	$0.585_{(0.025)}$	$0.872_{(0.059)}$	$0.791_{(0.078)}$	$0.757_{(0.028)}$	$0.671_{(0.009)}$	0.748
Masking+Sup [25]	$0.818_{(0.039)}$	$0.849_{(0.024)}$	$0.792_{(0.043)}$	$0.611_{(0.003)}$	$\underline{0.888}_{(0.038)}$	$\underline{0.819}_{(0.054)}$	$0.828_{(0.013)}$	$0.685_{(0.013)}$	0.786
Contextpred+Sup [25]	$0.831_{(0.018)}$	$0.852_{(0.039)}$	$0.765_{(0.061)}$	$\underline{0.612}_{(0.026)}$	$0.871_{(0.051)}$	$0.807_{(0.039)}$	$\underline{0.829}_{(0.026)}$	$0.687_{(0.008)}$	0.782
GraphLoG [58]	$0.766_{(0.040)}$	$0.799_{(0.025)}$	$0.526_{(0.101)}$	$0.583_{(0.016)}$	$0.867_{(0.071)}$	$0.747_{(0.063)}$	$0.691_{(0.041)}$	$0.608_{(0.011)}$	0.698
GraphCL [63]	$0.802_{(0.045)}$	$0.836_{(0.021)}$	$0.663_{(0.018)}$	$0.576_{(0.011)}$	$0.864_{(0.050)}$	$0.779_{(0.062)}$	$0.758_{(0.036)}$	$0.639_{(0.012)}$	0.740
JOAO [62]	$\underline{0.834}_{(0.033)}$	$0.854_{(0.023)}$	$0.747_{(0.039)}$	$0.612_{(0.007)}$	$0.854_{(0.061)}$	$0.800_{(0.062)}$	$0.801_{(0.023)}$	$0.653_{(0.014)}$	0.769
GROVER [44]	$0.809_{(0.046)}$	$\underline{0.885}_{(0.009)}$	$\underline{0.809}_{(0.048)}$	$0.591_{(0.030)}$	$0.870_{(0.048)}$	$0.818_{(0.056)}$	$0.827_{(0.019)}$	$\underline{0.708}_{(0.015)}$	$\underline{0.790}$
3DInfomax [46]	$0.810_{(0.044)}$	$0.806_{(0.037)}$	$0.743_{(0.014)}$	$0.585_{(0.021)}$	$0.852_{(0.054)}$	$0.784_{(0.072)}$	$0.738_{(0.021)}$	$0.609_{(0.021)}$	0.741
GraphMVP [32]	$0.741_{(0.012)}$	$0.836_{(0.028)}$	$0.696_{(0.051)}$	$0.563_{(0.023)}$	$0.833_{(0.071)}$	$0.669_{(0.057)}$	$0.778_{(0.022)}$	$0.645_{(0.006)}$	0.720
KPGT	$\mathbf{0.868}_{(0.011)}$	$\mathbf{0.896}_{(0.010)}$	$\mathbf{0.856}_{(0.028)}$	$\mathbf{0.644}_{(0.018)}$	$\mathbf{0.890}_{(0.048)}$	$\mathbf{0.885}_{(0.047)}$	$\mathbf{0.838}_{(0.020)}$	$\mathbf{0.727}_{(0.016)}$	**0.825**

Method	Classification dataset ↑								
	Transfer learning setting								
	BACE	BBBP	ClinTox	SIDER	Estrogen	MetStab	Tox21	ToxCast	**AVG**
Infomax [51]	$0.839_{(0.008)}$	$0.840_{(0.026)}$	$0.661_{(0.026)}$	$0.616_{(0.024)}$	$0.888_{(0.056)}$	$0.837_{(0.066)}$	$0.816_{(0.021)}$	$0.690_{(0.012)}$	0.773
Edgepred [20]	$0.817_{(0.034)}$	$0.873_{(0.016)}$	$0.730_{(0.017)}$	$0.603_{(0.025)}$	$0.881_{(0.061)}$	$0.844_{(0.054)}$	$0.818_{(0.025)}$	$0.712_{(0.011)}$	0.785
Masking [25]	$0.823_{(0.004)}$	$0.864_{(0.028)}$	$0.729_{(0.039)}$	$0.573_{(0.012)}$	$0.869_{(0.050)}$	$0.868_{(0.061)}$	$0.798_{(0.025)}$	$0.663_{(0.018)}$	0.773
Contextpred [25]	$0.840_{(0.009)}$	$0.877_{(0.026)}$	$0.732_{(0.017)}$	$0.609_{(0.013)}$	$0.882_{(0.065)}$	$0.857_{(0.053)}$	$0.806_{(0.012)}$	$0.714_{(0.018)}$	0.790
Masking+Sup [25]	$0.824_{(0.021)}$	$0.859_{(0.020)}$	$0.796_{(0.079)}$	$0.606_{(0.005)}$	$0.888_{(0.041)}$	$0.872_{(0.051)}$	$0.827_{(0.021)}$	$0.715_{(0.007)}$	0.799
Contextpred+Sup [25]	$\underline{0.855}_{(0.023)}$	$0.875_{(0.011)}$	$0.802_{(0.079)}$	$0.620_{(0.009)}$	$0.885_{(0.053)}$	$0.859_{(0.055)}$	$\underline{0.840}_{(0.023)}$	$\underline{0.724}_{(0.015)}$	0.807
GraphLoG [58]	$0.830_{(0.014)}$	$0.846_{(0.008)}$	$0.667_{(0.021)}$	$0.615_{(0.013)}$	$0.871_{(0.054)}$	$0.850_{(0.080)}$	$0.796_{(0.025)}$	$0.677_{(0.019)}$	0.769
GraphCL [63]	$0.825_{(0.018)}$	$0.887_{(0.019)}$	$0.691_{(0.065)}$	$0.587_{(0.026)}$	$0.875_{(0.048)}$	$0.821_{(0.066)}$	$0.805_{(0.017)}$	$0.696_{(0.023)}$	0.774
JOAO [62]	$0.826_{(0.029)}$	$0.879_{(0.020)}$	$0.741_{(0.047)}$	$\underline{0.640}_{(0.010)}$	$0.861_{(0.066)}$	$0.837_{(0.058)}$	$0.823_{(0.022)}$	$0.711_{(0.014)}$	0.790
GROVER [44]	$0.840_{(0.030)}$	$\underline{0.887}_{(0.006)}$	$0.874_{(0.048)}$	$0.638_{(0.005)}$	$\underline{0.892}_{(0.044)}$	$\underline{0.876}_{(0.038)}$	$0.838_{(0.017)}$	$0.696_{(0.014)}$	$\underline{0.818}$
3DInfomax [46]	$0.811_{(0.048)}$	$0.877_{(0.014)}$	$\underline{0.887}_{(0.033)}$	$0.585_{(0.017)}$	$0.880_{(0.054)}$	$0.866_{(0.047)}$	$0.805_{(0.032)}$	$0.716_{(0.013)}$	0.804
GraphMVP [32]	$0.818_{(0.012)}$	$0.860_{(0.034)}$	$0.719_{(0.044)}$	$0.584_{(0.026)}$	$0.865_{(0.061)}$	$0.820_{(0.066)}$	$0.799_{(0.018)}$	$0.689_{(0.010)}$	0.769
KPGT	$\mathbf{0.855}_{(0.011)}$	$\mathbf{0.908}_{(0.010)}$	$\mathbf{0.946}_{(0.022)}$	$\mathbf{0.649}_{(0.009)}$	$\mathbf{0.905}_{(0.028)}$	$\mathbf{0.889}_{(0.047)}$	$\mathbf{0.848}_{(0.013)}$	$\mathbf{0.746}_{(0.002)}$	**0.843**

tried 48 hyper-parameter combinations to find the best results for each task. For more training details please refer to Appendix A.2.2.

Evaluation Protocols. The downstream molecular property prediction tests were performed under two evaluation protocols, that is, the feature extraction setting and the transfer learning setting. For the feature extraction setting, we first fixed the pre-trained model and used it as a feature extractor to obtain the molecular graph representations of data samples, and then trained the predictor to make predictions. For the transfer learning setting, we fine-tuned all the parameters in the model.

Baselines. We comprehensively evaluated KPGT against twelve state-of-the-art self-supervised learning methods on molecular graphs, including six generative methods, i.e., Edgepred [20], Masking [25], Contextpred [25], Masking+Sup [25], Contextpred+Sup [25] and GROVER [44], and six contrastive methods, including Infomax [51], GraphLoG [58], GraphCL [63], JOAO[62], 3DInfomax [46] and GraphMVP [32].

4.2 Results on Downstream Tasks

We reported the evaluation results of molecular property prediction under both feature extraction and transfer learning settings in Tables 1 and 2. The comprehensive test results suggested the following trends:

Observation 1. From the test results, KPGT outperformed baseline methods with considerable margins on all classification and regression datasets under both feature extraction and transfer learning settings. For the feature extraction setting, the overall relative improvement was 5% on all datasets, with 3.5% on classification datasets and 8.9% on regression datasets. For the transfer learning setting, the overall relative improvement was 3.9% on all datasets, with 2.2% on classification datasets and 8.3% on regression datasets. This significant improvement demonstrated that KPGT can provide a useful tool for accurate molecular property prediction.

Table 2: The RMSE performance of different methods on the regression datasets under both feature extraction and transfer learning settings. The lower result is better (marked by ↓). AVG represents the averaging results over all the datasets. The numbers in brackets are the standard deviations. The best result for each dataset is marked in bold and the second-best result is underlined.

Method	Regression dataset ↓							
	Feature extraction setting				Transfer learning setting			
	FreeSolv	ESOL	Lipo	**AVG**	FreeSolv	ESOL	Lipo	**AVG**
Infomax [51]	$4.119_{(0.974)}$	$1.462_{(0.076)}$	$0.978_{(0.076)}$	2.186	$3.416_{(0.928)}$	$1.096_{(0.116)}$	$0.799_{(0.047)}$	1.770
Edgepred [20]	$3.849_{(0.950)}$	$2.272_{(0.213)}$	$1.030_{(0.024)}$	2.384	$3.076_{(0.585)}$	$1.228_{(0.073)}$	$0.719_{(0.013)}$	1.674
Masking [25]	$3.646_{(0.947)}$	$2.100_{(0.040)}$	$1.063_{(0.028)}$	2.270	$3.040_{(0.334)}$	$1.326_{(0.115)}$	$0.724_{(0.012)}$	1.697
Contextpred [25]	$3.141_{(0.905)}$	$1.349_{(0.069)}$	$0.969_{(0.076)}$	1.820	$2.890_{(1.077)}$	$1.077_{(0.029)}$	$0.722_{(0.034)}$	1.563
Masking+Sup [25]	$3.210_{(0.876)}$	$1.387_{(0.007)}$	$\underline{0.725}_{(0.033)}$	1.774	$2.883_{(0.559)}$	$1.297_{(0.114)}$	$0.681_{(0.026)}$	1.620
Contextpred+Sup [25]	$3.105_{(0.701)}$	$1.477_{(0.038)}$	$0.754_{(0.032)}$	1.779	$\underline{2.383}_{(0.624)}$	$1.178_{(0.026)}$	$0.681_{(0.030)}$	1.414
GraphLoG [58]	$4.174_{(1.077)}$	$2.335_{(0.073)}$	$1.104_{(0.024)}$	2.537	$2.961_{(0.847)}$	$1.249_{(0.010)}$	$0.780_{(0.020)}$	1.663
GraphCL [63]	$4.014_{(1.361)}$	$1.835_{(0.111)}$	$0.945_{(0.024)}$	2.264	$3.149_{(0.273)}$	$1.540_{(0.086)}$	$0.777_{(0.034)}$	1.822
JOAO [62]	$3.466_{(1.114)}$	$1.771_{(0.053)}$	$0.933_{(0.027)}$	2.056	$3.950_{(1.202)}$	$1.220_{(0.028)}$	$0.710_{(0.031)}$	1.960
GROVER [44]	$2.991_{(1.052)}$	$\underline{0.928}_{(0.027)}$	$0.752_{(0.010)}$	<u>1.557</u>	$2.385_{(1.047)}$	$0.986_{(0.126)}$	$\underline{0.625}_{(0.006)}$	<u>1.332</u>
3DInfomax [46]	$2.919_{(0.243)}$	$1.906_{(0.246)}$	$1.045_{(0.040)}$	1.957	$2.639_{(0.772)}$	$\underline{0.891}_{(0.131)}$	$0.671_{(0.033)}$	1.400
GraphMVP [32]	$\underline{2.532}_{(0.247)}$	$1.937_{(0.147)}$	$0.990_{(0.024)}$	1.819	$2.874_{(0.756)}$	$1.355_{(0.038)}$	$0.712_{(0.025)}$	1.647
KPGT	$\mathbf{2.314}_{(0.841)}$	$\mathbf{0.848}_{(0.103)}$	$\mathbf{0.656}_{(0.023)}$	**1.273**	$\mathbf{2.121}_{(0.837)}$	$\mathbf{0.803}_{(0.008)}$	$\mathbf{0.600}_{(0.010)}$	**1.175**

Observation 2. Methods incorporating additional knowledge of molecules generally achieved better performance. Among the baseline methods, Contextpred+Sup, Masking+Sup and GROVER incorporated additional knowledge of molecules by introducing graph-level pre-training tasks, that is, learning to predict specific properties of molecules, e.g., bio-activities and motifs, using the graph representations. Although their node-level pre-training tasks (i.e., predicting the masked nodes or subgraphs) were ill-defined, these methods still generally outperformed other baseline methods that did not exploit additional knowledge, which indicated that incorporating additional knowledge was important to the success of the self-supervised learning methods on molecular graphs.

Observation 3. Methods employing high-capacity backbone networks achieved better performance. Among all the baseline methods, GROVER also employed a transformer-based model with around 100 million parameters as its backbone network and achieved superior performance in comparison with other baseline methods. This result further validated our point that a high-capacity backbone network was necessary to yield promising results for self-supervised learning on molecular graphs.

Observation 4. The methods under the transfer learning setting generally achieved better performance over those under the feature extraction setting. This result was reasonable because fine-tuning all the parameters enabled the models to capture more task-specific information. However, on some datasets, KPGT under the transfer learning setting only achieved comparable (i.e., on the SIDER and MetStab datasets) or even worse (i.e., on the BACE dataset) performance compared to that under the feature extraction setting, which may be caused by the catastrophic forgetting problem in the fine-tuning [27, 30]. Therefore, exploring different fine-tuning strategies is a potential future direction to further improve the performance of molecular property prediction.

Table 3: Comparison results of different backbone networks on classification and regression datasets, measured in terms of AUROC and RMSE, respectively, under the transfer learning setting.

Backbone	Classification dataset ↑	Regression dataset ↓
	AVG	**AVG**
GIN [57]	0.817	1.229
Vanilla Transformer	0.822	1.260
Graphormer [61]	0.826	1.189
LiGhT	0.829	1.162

4.3 Ablation Studies

4.3.1 How Powerful is the LiGhT Backbone Network? To evaluate the expressive power of LiGhT, we replaced LiGhT in KPGT with other backbone networks, including GIN, vanilla Transformer (i.e., without path encoding and distance encoding) and Graphormer [61]. For a fair comparison, we restricted all the models to have nearly the same number of parameters (i.e., 3.5 million). Then we individually pre-trained the models with our self-supervised learning strategy. As shown in Table 3, our backbone network LiGhT consistently outperformed GIN, vanilla Transformer and Graphormer, which verified that LiGhT can provide a reliable backbone network for self-supervised learning on molecular graphs.

4.3.2 How Powerful is the Knowledge-Guided Pre-training Strategy? To evaluate our self-supervised learning strategy, we first compared the prediction performances of the pre-trained LiGhT (KPGT) and the LiGhT without pre-training, both of which followed the same hyper-parameter settings. As shown in Table 4, the pre-trained model consistently achieved superior prediction performance and

Table 4: Comparison results of the pre-trained LiGhT (KPGT) and LiGhT (i.e., without pre-training) on classification and regression datasets in terms of AUROC and RMSE, respectively, under the transfer learning setting.

	Classification dataset ↑	Regression dataset ↓
	AVG	**AVG**
LiGhT	0.803	1.473
KPGT	0.843	1.175
Relative improvement	5.1%	18.8%

Table 5: Comparison results between KPGT and baseline methods, which also incorporated the same additional knowledge, on classification and regression datasets, measured in terms of AUROC and RMSE, respectively, under the feature extraction setting.

	Classification dataset ↑	Regression dataset ↓
	AVG	**AVG**
MD+FP	0.776	1.698
GROVER+FP	0.789	1.541
KPGT	0.825	1.273

the relative improvements were 5.1% and 18.8% on classification and regression tasks, respectively.

Next, we asked whether simply using the same additional knowledge employed in KPGT could provide better results. We introduced two baseline methods, which also incorporated the same additional knowledge employed in our framework. First, we directly concatenated the molecular descriptors and fingerprints employed in our framework as input and then passed through a 2-layer MLP for prediction (denoted by MD+FP). Note that the previous method GROVER [44] had already concatenated the molecular descriptors employed in our framework to the graph representation derived from the readout function. Here, we further concatenated the fingerprints to such output and passed through a 2-layer MLP for the final prediction (denoted by GROVER+FP). For a fair comparison, we fixed the parameters of the feature extractors in both GROVER+FP and KPGT (i.e., the feature extraction setting). As shown in Table 5, KPGT still outperformed the baseline methods. These results confirmed that our self-supervised learning strategy can potentially learn the abundant semantics of molecules beyond the introduced additional knowledge.

4.3.3 Effect of Different Masking Rates. In our self-supervised learning strategy, we masked a proportion of the nodes and additional knowledge in the molecular line graph. Here, we evaluated the effect of different masking rates. In particular, we individually pre-trained KPGT with the masking rates of 15%, 30%, 50% and 60%, and reported the corresponding prediction results on the downstream tasks (see Table 6). The results showed that setting the masking rate to 50% achieved the best prediction performance. Such an optimal masking rate was much larger than that equipped in the previous methods (i.e., 15%) [25, 44], which indicated that the additional knowledge incorporated in KPGT accurately guided the model to predict the masked nodes and thus enabled it to capture the semantics of molecules. Moreover, recent research in the field of CV also has shown that larger masking rates in the pre-training resulted in better results on the downstream tasks [21].

Table 6: Ablation results of KPGT with different masking rates on classification and regression datasets, measured in terms of AUROC and RMSE, respectively, under the transfer learning setting.

Masking rate	Classification dataset ↑	Regression dataset ↓
	AVG	**AVG**
15%	0.827	1.228
30%	0.835	1.216
50%	0.843	1.175
60%	0.832	1.186

5 CONCLUSION

In this paper, we first point out the two major issues of current self-supervised learning methods on molecular graphs, that is, the ill-defined pre-training tasks and the limited model capacity. Next, we propose a novel self-supervised learning framework, named Knowledge-guided Pre-training of Graph Transformer (KPGT), which consists of two main components, that is, the Line Graph Transformer (LiGhT) and a knowledge-guided pre-training strategy, to alleviate the aforementioned issues and learn more generalizable, transferable and robust representations of molecular graphs to improve the performance of the downstream molecular property prediction tasks. Extensive test results demonstrated that KPGT can offer superior performance over current state-of-the-art methods on several molecular property prediction datasets.

ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Program of China (2021YFF1201300), the National Natural Science Foundation of China (61872216, T2125007 to JZ, 31900862 to DZ), the Turing AI Institute of Nanjing, and the Tsinghua-Toyota Joint Research Fund.

REFERENCES

[1] 2017. Tox21 Challenge. (2017). http://tripod.nih.gov/tox21/challenge/
[2] Abien Fred Agarap. 2018. Deep Learning using Rectified Linear Units (ReLU). *CoRR* abs/1803.08375 (2018).
[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* (2016).
[4] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. 2021. Directional graph networks. In *International Conference on Machine Learning*. PMLR, 748–758.
[5] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. 2018. Machine learning for molecular and materials science. *Nature* 559, 7715 (2018), 547–555.
[6] Adrià Cereto-Massagué et al. 2015. Molecular fingerprint similarity search in virtual screening. *Methods* 71 (2015), 58–63.
[7] Benson Chen, Regina Barzilay, and Tommi S. Jaakkola. 2019. Path-Augmented Graph Transformer Network. *CoRR* abs/1905.12712 (2019).
[8] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. 2020. Principal Neighbourhood Aggregation for Graph Nets. In *NeurIPS 2020*.

[9] John S Delaney. 2004. ESOL: estimating aqueous solubility directly from molecular structure. *Journal of chemical information and computer sciences* 44, 3 (2004), 1000–1005.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT 2019*. 4171–4186.

[11] Jie Dong, Ning-Ning Wang, Zhi-Jiang Yao, Lin Zhang, Yan Cheng, Defang Ouyang, Ai-Ping Lu, and Dong-Sheng Cao. 2018. ADMETlab: a platform for systematic ADMET evaluation based on a comprehensively collected ADMET database. *Journal of cheminformatics* 10, 1 (2018), 1–11.

[12] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, et al. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research* 22, 107 (2021), 1–48.

[13] Luciano Floridi and Massimo Chiriatti. 2020. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines* 30, 4 (2020), 681–694.

[14] Anna Gaulton et al. 2017. The ChEMBL database in 2017. *Nucleic acids research* 45, D1 (2017), D945–D954.

[15] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. 2012. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic acids research* 40, D1 (2012), D1100–D1107.

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

[17] 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*. PMLR, 1263–1272.

[18] Landrum Greg et al. 2021. *rdkit/rdkit: 2021_09_2 (Q3 2021) Release*. https://doi.org/10.5281/zenodo.5589557

[19] Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. *Exploring network structure, dynamics, and function using NetworkX*. Technical Report. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

[20] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS 2017*. 1024–1034.

[21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2021. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377* (2021).

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. 2021. Masked Autoencoders Are Scalable Vision Learners. *CoRR* abs/2111.06377 (2021).

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR 2020*. 9726–9735.

[24] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).

[25] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR 2020*.

[26] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR 2015*.

[27] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[28] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking graph transformers with spectral attention. *NeurIPS 2021* 34 (2021).

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

[30] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. 2019. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *ICML 2019*. PMLR, 3925–3934.

[31] Shengchao Liu, Mehmet Furkan Demirel, and Yingyu Liang. 2019. N-Gram Graph: Simple Unsupervised Representation for Graphs, with Applications to Molecules. In *NeurIPS 2019*. 8464–8476.

[32] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. 2021. Pre-training Molecular Graph Representation with 3D Geometry. *arXiv preprint arXiv:2110.07728* (2021).

[33] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. 2021. Swin Transformer V2: Scaling Up Capacity and Resolution. *arXiv preprint arXiv:2111.09883* (2021).

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

[35] Lukasz Maziarka, Tomasz Danel, Slawomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanislaw Jastrzebski. 2020. Molecule Attention Transformer. *CoRR* abs/2002.08264 (2020).

[36] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. 2021. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667* (2021).

[37] David L Mobley and J Peter Guthrie. 2014. FreeSolv: a database of experimental and calculated hydration free energies, with input files. *Journal of computer-aided molecular design* 28, 7 (2014), 711–720.

[38] Hirotomo Moriwaki, Yu-Shi Tian, Norihito Kawashita, and Tatsuya Takagi. 2018. Mordred: a molecular descriptor calculator. *Journal of cheminformatics* 10, 1 (2018), 1–14.

[39] Adam Paszke et al. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[40] Sabina Podlewska and Rafał Kafel. 2018. MetStabOn—online platform for metabolic stability predictions. *International journal of molecular sciences* 19, 4 (2018), 1040.

[41] Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. 2019. *Deep Learning for the Life Sciences*. O’Reilly Media.

[42] Ann M Richard et al. 2016. ToxCast chemical landscape: paving the road to 21st century toxicology. *Chemical research in toxicology* 29, 8 (2016), 1225–1251.

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

[44] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-Supervised Graph Transformer on Large-Scale Molecular Data. In *NeurIPS 2020*.

[45] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL-HLT 2018*. Association for Computational Linguistics, 464–468.

[46] Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Liò. 2021. 3D Infomax improves GNNs for Molecular Property Prediction. *arXiv preprint arXiv:2110.04126* (2021).

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

[48] Alexandre Tkatchenko. 2020. Machine learning for chemical discovery. *Nature Communications* 11, 1 (2020), 1–4.

[49] Han Van De Waterbeemd and Eric Gifford. 2003. ADMET in silico modelling: towards prediction paradise? *Nature reviews Drug discovery* 2, 3 (2003), 192–204.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS 2017*. 5998–6008.

[51] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR 2019*. OpenReview.net.

[52] Minjie Wang et al. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv preprint arXiv:1909.01315* (2019).

[53] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. 2021. MolCLR: molecular contrastive learning of representations via graph neural networks. *arXiv preprint arXiv:2102.10056* (2021).

[54] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. 2020. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies* 37 (2020), 1–12.

[55] 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. *Chemical science* 9, 2 (2018), 513–530.

[56] Zhaoping Xiong et al. 2019. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry* 63, 16 (2019), 8749–8760.

[57] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR 2019*.

[58] Minghao Xu, Hang Wang, Bingbing Ni, Hongyu Guo, and Jian Tang. 2021. Self-supervised Graph-level Representation Learning with Local and Global Structure. In *ICML 2021*, Vol. 139. 11548–11558.

[59] Ling Xue and Jurgen Bajorath. 2000. Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening. *Combinatorial chemistry & high throughput screening* 3, 5 (2000), 363–372.

[60] Kevin Yang et al. 2019. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling* 59, 8 (2019), 3370–3388.

[61] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Badly for Graph Representation?. In *NeurIPS 2021*.

[62] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph Contrastive Learning Automated. In *ICML 2021*, Vol. 139. 12121–12132.

[63] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph Contrastive Learning with Augmentations. In *NeurIPS 2020*.

[64] Jie Zhou et al. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

A APPENDIX

A.1 The Ill-Defined Pre-Training Task

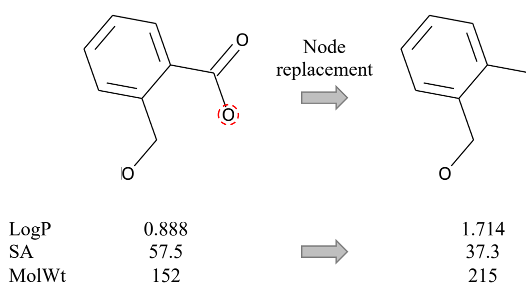

Figure 4: A small modification of a molecule can lead to great changes in its raw properties.

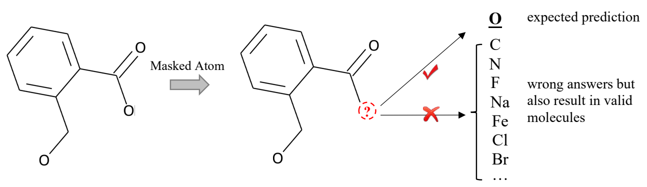

Figure 5: The dependence between the masked node and its neighboring nodes is weak.

A.2 Implementation Details

Table 7: Atom features.

Feature	Size	Description
atom type	101	type of atoms (e.g., C, N, O), by atomic number (one-hot)
degree	12	number of bonds the atom is involved in (one-hot)
formal charge	1	electronic charge assigned to the atom (integer)
chiral center	1	is charity center (integer)
chirality type	2	R or S (one-hot)
number of H	6	number of bonded hydrogen atoms (one-hot)
atomic mass	1	mass of the atom, divided by 100 (integer)
aromaticity	1	whether this atom is part of an aromatic system (integer)
radical electrons	6	number of radical electrons (one-hot)
hybridization	6	sp, sp^2, sp^3, sp^3d, sp^3d^2 or unknown (one-hot)

Table 8: Bond features.

Feature	Size	Description
bond type	5	type of bonds (i.e., single, double, triple, aromatic or unknown) (one-hot)
stereo	7	StereoNone, StereoAny, StereoZ, StereoE, StereoCis, StereoTrans or unknown(one-hot)
in ring	1	whether the bond is part of a ring (integer)
conjugated	1	whether the bond is conjugated (integer)

A.2.1 Molecular Graph Construction. We abstracted the molecular graph and initialized the features of nodes and edges in the molecular graphs via RDKit [18]. Tables 7 and 8summarize the atom and bond features used in our experiments, respectively.

Table 9: The pre-training and fine-tuning hyper-parameters for KPGT.

Hyper-parameter	Pre-training	Fine-tuning
hidden size	768	768
number of layers	12	12
number of attention heads	12	12
dropout rate	0.1	[0, 0.05, 0.1, 0.2]
batch size	1024	32
learning rate	$2e^{-4}$	$[1e^{-6}, 3e^{-6}, 1e^{-5}, 3e^{-5}]$
learning rate decay	polynomial	polynomial
weight decay	$1e^{-6}$	$[0, 1e^{-6}, 1e^{-4}]$
Gradient clipping	5.0	5.0
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999
predictor hidden size	NA	256
predictor layer	NA	2
mask rate	0.5	NA

A.2.2 Training Details. All models were implemented in PyTorch [39] version 1.10.0 and DGL [52] version 0.7.2 with CUDA version 11.3 and Python 3.7.

The pre-training and fine-tuning hyper-parameters for KPGT are summarized in Table 9. For KPGT and baseline methods, we tried 48 hyper-parameter combinations of different learning rates, dropout rates, and weight decay listed in Table 9 via grid search to find the best results for each dataset.

A.3 Datasets

The detailed information of the molecular property datasets used in our experiments is summarized in Table 10. The details of each molecular property dataset are listed below:

- BACE is a collection of molecules that can act as the inhibitors of human β-secretase 1 (BACE-1) [47].
- BBBP is a dataset that records whether a molecule can penetrate the blood-brain barrier [34].

Table 10: Details of datasets.

Dataset	Type	#Tasks	#Molecules
BACE	Classification	1	1513
BBBP	Classification	1	2039
ClinTox	Classification	2	1478
SIDER	Classification	27	1427
ToxCast	Classification	617	8575
Tox21	Classification	12	7831
Estrogen	Classification	2	3122
MetStab	Classification	2	2267
FreeSolv	Regression	1	642
ESOL	Regression	1	1128
Lipophilicity	Regression	1	4200

- ClinTox is a collection of drugs approved through the Food and Drug Administration (FDA) and eliminated due to the toxicity during clinical trials [16].
- SIDER records the adverse drug reactions of marked drugs [29].
- ToxCast contains multiple toxicity labels for molecules obtained through high-throughput screening tests [42].
- Tox21 is a public database containing the toxicity of compounds [1].
- Estrogen contains the molecules with known activities towards the estrogen receptors extracted from the ChEMBL dataset [15].
- MetStab is a dataset measuring the half-life time of molecules within an organism [40].
- ESOL is a dataset recording the solubility of compounds [9].
- Lipophilicity is a dataset measuring the molecular membrane permeability and solubility [15].
- FreeSolv contains the hydration free energy of small molecules in water from both experiments and alchemical free energy calculation [37].