Self-Supervised Graph Transformer on Large-Scale Molecular Data

Yu Rong[1]*, **Yatao Bian**[1]*, **Tingyang Xu**[1], **Weiyang Xie**[1], **Ying Wei**[1],
Wenbing Huang[2]†, **Junzhou Huang**[1]
[1]Tencent AI Lab
[2] Beijing National Research Center for Information Science and Technology (BNRist),
Department of Computer Science and Technology, Tsinghua University
yu.rong@hotmail.com, yatao.bian@gmail.com, {tingyangxu, weiyangxie}@tencent.com
judyweiying@gmail.com, hwenbing@126.com, jzhuang@uta.edu

Abstract

How to obtain informative representations of molecules is a crucial prerequisite in AI-driven drug design and discovery. Recent researches abstract molecules as graphs and employ Graph Neural Networks (GNNs) for molecular representation learning. Nevertheless, two issues impede the usage of GNNs in real scenarios: (1) insufficient labeled molecules for supervised training; (2) poor generalization capability to new-synthesized molecules. To address them both, we propose a novel framework, GROVER, which stands for **G**raph **R**epresentation fr**O**m self-super**V**ised m**E**ssage passing t**R**ansformer. With carefully designed self-supervised tasks in node-, edge- and graph-level, GROVER can learn rich structural and semantic information of molecules from enormous unlabelled molecular data. Rather, to encode such complex information, GROVER integrates Message Passing Networks into the Transformer-style architecture to deliver a class of more expressive encoders of molecules. The flexibility of GROVER allows it to be trained efficiently on large-scale molecular dataset without requiring any supervision, thus being immunized to the two issues mentioned above. We pre-train GROVER with 100 million parameters on 10 million unlabelled molecules—the biggest GNN and the largest training dataset in molecular representation learning. We then leverage the pre-trained GROVER for molecular property prediction followed by task-specific fine-tuning, where we observe a huge improvement (more than 6% on average) from current state-of-the-art methods on 11 challenging benchmarks. The insights we gained are that well-designed self-supervision losses and largely-expressive pre-trained models enjoy the significant potential on performance boosting.

1 Introduction

Inspired by the remarkable achievements of deep learning in many scientific domains, such as computer vision [19, 57], natural language processing [51, 53], and social networks [3, 31], researchers are exploiting deep learning approaches to accelerate the process of drug discovery and reduce costs by facilitating the rapid identification of molecules [5]. Molecules can be naturally represented by molecular graphs which preserve rich structural information. Therefore, supervised deep learning of graphs, especially with Graph Neural Networks(GNNs) [25, 46] have shown promising results in many tasks, such as molecular property prediction [13, 23] and virtual screening [56, 64].

*Equal contribution.
†Wenbing Huang is the corresponding author.

Despite the fruitful progress, two issues still impede the usage of deep learning in real scenarios: (1) insufficient labeled data for molecular tasks; (2) poor generalization capability of models in the enormous chemical space. Different from other domains (such as image classification) that have rich-source labeled data, getting labels of molecular property requires wet-lab experiments which is time-consuming and resource-costly. As a consequence, most public molecular benchmarks contain far-from-adequate labels. Conducting deep learning on these benchmarks is prone to over-fitting and the learned model can hardly cope with the out-of-distribution molecules.

Indeed, it has been a long-standing goal in deep learning to improve the generalization power of neural networks. Towards this goal, certain progress has been made. For example, in Natural Language Processing (NLP), researchers can pre-train the model from large-scale unlabeled sentences via a newly-proposed technique—the self-supervised learning. Several successful self-supervised pretraining strategies, such as BERT [9] and GPT [38] have been developed to tackle a variety of language tasks. By contending that molecule can be transformed into sequential representation—SMILES [59], the work by [58] tries to adopt the BERT-style method to pretrain the model, and Liu et.al. [29] also exploit the idea from N-gram approach in NLP and conducts vertices embedding by predicting the vertices attributes. Unfortunately, these approaches fail to explicitly encode the structural information of molecules as using the SMILES representation is not topology-aware.

Without using SMILES, several works aim to establish a pre-trained model directly on the graph representations of molecules. Hu et.al. [18] investigate the strategies to construct the three pre-training tasks, i.e., context prediction and node masking for node-level self-supervised learning and graph property prediction for graph-level pre-training. We argue that the formulation of pre-training in this way is suboptimal. First, in the masking task, they treat the atom type as the label. Different from NLP tasks, the number of atom types in molecules is much smaller than the size of a language vocabulary. Therefore, it would suffer from serious representation ambiguity and the model is hard to encode meaningful information especially for the highly frequent atoms. Second, the graph-level pre-training task in [18] is supervised. This limits the usage in practice since most of molecules are completely unlabelled, and it also introduces the risk of *negative transfer* for the downstream tasks if they are inconsistent to the graph-level supervised loss.

In this paper, we improve the pre-training model for molecular graph by introducing a novel molecular representation framework, GROVER, namely, **G**raph **R**epresentation fr**O**m self-super**V**ised m**E**ssage passing t**R**ansformer. GROVER constructs two types of self-supervised tasks. For the node/edge-level tasks, instead of predicting the node/edge type alone, GROVER randomly masks a local subgraph of the target node/edge and predicts this contextual property from node embeddings. In this way, GROVER can alleviate the ambiguity problem by considering both the target node/edge and its context being masked. For the graph-level tasks, by incorporating the domain knowledge, GROVER extracts the semantic motifs existing in molecular graphs and predicts the occurrence of these motifs for a molecule from graph embeddings. Since the semantic motifs can be obtained by a low-cost pattern matching method, GROVER can make use of any molecular to optimize the graph-level embedding. With self-supervised tasks in node-, edge- and graph-level, GROVER can learn rich structural and semantic information of molecules from enormous unlabelled molecular data. Rather, to encode such complex information, GROVER integrates Message Passing Networks with the Transformer-style architecture to deliver a class of highly expressive encoders of molecules. The flexibility of GROVER allows it to be trained efficiently on large-scale molecular data without requiring any supervision. We pre-train GROVER with 100 million parameters on 10 million of unlabelled molecules—the biggest GNN and the largest training dataset that have been applied. We then leverage the pre-trained GROVER models to downstream molecular property prediction tasks followed by task-specific fine-tuning. On the downstream tasks, GROVER achieve 22.4% relative improvement compared with [29] and 7.4% relative improvement compared with [18] on classification tasks. Furthermore, even compared with current state-of-the-art results for each data set, we observe a huge relative improvement of GROVER (more than 6% on average) over 11 popular benchmarks.

2 Related Work

Molecular Representation Learning. To represent molecules in the vector space, the traditional chemical fingerprints, such as ECFP [42], try to encode the neighbors of atoms in the molecule into a fix-length vector. To improve the expressive power of chemical fingerprints, some studies [7, 10] introduce convolutional layers to learn the neural fingerprints of molecules, and apply the neural fingerprints to the downstream tasks, such as property prediction. Following these works, [21, 62] take the SMILES representation [59] as input and use RNN-based models to produce molecular representations. Recently, many works [23, 45, 46] explore the graph convolutional network to encode molecular graphs into neural fingerprints. A slot of work [44, 61] propose to learn the aggregation weights by extending the Graph Attention Network (GAT) [54]. To better capture the interactions among atoms, [13] proposes to use a message passing framework and [25, 63] extend this framework to model bond interactions. Furthermore, [30] builds a hierarchical GNN to capture multilevel interactions.

Self-supervised Learning on Graphs. Self-supervised learning has a long history in machine learning and has achieved fruitful progresses in many areas, such as computer vision [35] and language modeling [9]. The traditional graph embedding methods [14, 37] define different kinds of graph proximity, i.e., the vertex proximity relationship, as the self-supervised objective to learn vertex embeddings. GraphSAGE [15] proposes to use a random-walk based proximity objective to train GNN in an unsupervised fashion. [36, 50, 55] exploit the mutual information maximization scheme to construct objective for GNNs. Recently, two works are proposed to construct unsupervised representations for molecular graphs. Liu et.al. [29] employ an N-gram model to extract the context of vertices and construct the graph representation by assembling the vertex embeddings in short walks in the graph. Hu et.al. [18] investigate various strategies to pre-train the GNNs and propose three self-supervised tasks to learn molecular representations. However, [18] isolates the highly correlated tasks of context prediction and node/edge type prediction, which makes it difficult to preserve domain knowledge between the local structure and the node attributes. Besides, the graph-level task in [18] is constructed by the supervised property labels, which is impeded by the limited number of supervised labels of molecules and has demonstrated the negative transfer in the downstream tasks. Contrast with [18], the molecular representations derived by our method are more appropriate in terms of persevering the domain knowledge, which has demonstrated remarkable effectiveness in downstream tasks without negative transfer.

3 Preliminaries of Transformer-style Models and Graph Neural Networks

We briefly introduce the concepts of supervised graph learning, Transformer [53], and GNNs in this section.

Supervised learning tasks of graphs. A molecule can be abstracted as a graph $G = (\mathcal{V}, \mathcal{E})$, where $|\mathcal{V}| = n$ refers to a set of n nodes (atoms) and $|\mathcal{E}| = m$ refers to a set of m edges (bonds) in the molecule. $\mathcal{N}_v$ is used to denote the set of neighbors of node v. We use $\mathbf{x}_v$ to represent the initial features of node v, and $\mathbf{e}_{uv}$ as the initial features of edge (u, v). In graph learning, there are usually two categories of supervised tasks: i) *Node classification/regression*, where each node v has a label/target y_v, and the task is to learn to predict the labels of unseen nodes; ii) *Graph classification/regression*, where a set of graphs $\{G_1, ..., G_N\}$ and their labels/targets $\{y_1, ..., y_N\}$ are given, and the task is to predict the label/target of a new graph.

Attention mechanism and the Transformer-style architectures. The attention mechanism is the main building block of Transformer. We focus on multi-head attention, which stacks several scaled dot-product attention layers together and allows parallel running. One scaled dot-product attention layer takes a set of queries, keys, values $(\mathbf{q}, \mathbf{k}, \mathbf{v})$ as inputs. Then it computes the dot products of the query with all keys, and applies a softmax function to obtain the weights on the values. By stacking the set of $(\mathbf{q}, \mathbf{k}, \mathbf{v})$s into matrices $(\mathbf{Q}, \mathbf{K}, \mathbf{V})$, it admits highly optimized matrix multiplication operations. Specifically, the outputs can be arranged as a matrix:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\mathbf{Q}\mathbf{K}^\top / \sqrt{d})\mathbf{V}, \tag{1}$$

where d is the dimension of $\mathbf{q}$ and $\mathbf{k}$. Suppose we arrange k attention layers into the multi-head attention, then its output matrix can be written as,

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\text{head}_1, ..., \text{head}_k)\mathbf{W}^O, \\ \text{head}_i &= \text{Attention}(\mathbf{Q}\mathbf{W}_i^{\mathbf{Q}}, \mathbf{K}\mathbf{W}_i^{\mathbf{K}}, \mathbf{V}\mathbf{W}_i^{\mathbf{V}}), \end{aligned} \tag{2}$$

where $\mathbf{W}_i^{\mathbf{Q}}, \mathbf{W}_i^{\mathbf{K}}, \mathbf{W}_i^{\mathbf{V}}$ are the projection matrices of head i.

Graph Neural Networks (GNNs). Recently, GNNs have received a surge of interest in various domains, such as knowledge graph, social networks and drug discovery. The key operation of

GNNs lies in a message passing process, which involves message passing (also called neighborhood aggregation) between the nodes in the graph. The message passing operation iteratively updates a node v's hidden states, $\mathbf{h}_v$, by aggregating the hidden states of v's neighboring nodes and edges. In general, the message passing process involves several iterations, each iteration can be further partitioned into several hops. Suppose there are L iterations, and iteration l contains K_l hops. Formally, in iteration l, the k-th hop can be formulated as,

$$\begin{aligned}\mathbf{m}_v^{(l,k)} &= \text{AGGREGATE}^{(l)}(\{(\mathbf{h}_v^{(l,k-1)}, \mathbf{h}_u^{(l,k-1)}, \mathbf{e}_{uv}) \mid u \in \mathcal{N}_v\}), \\ \mathbf{h}_v^{(l,k)} &= \sigma(\mathbf{W}^{(l)}\mathbf{m}_v^{(l,k)} + \mathbf{b}^{(l)}),\end{aligned} \tag{3}$$

where $\mathbf{m}_v^{(l,k)}$ is the aggregated message, and $\sigma(\cdot)$ is some activation function. We make the convention that $\mathbf{h}_v^{(l,0)} := \mathbf{h}_v^{(l-1,K_{l-1})}$. There are several popular ways of choosing AGGREGATE$^{(l)}(\cdot)$, such as mean, max pooling and graph attention mechanism [15, 54]. For one iteration of message passing, there are a layer of trainable parameters (i.e., parameters inside AGGREGATE$^{(l)}(\cdot)$, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$. These parameters are shared across the K_l hops within the iteration l. After L iterations of message passing, the hidden states of the last hop in the last iteration are used as the embeddings of the nodes, i.e., $\mathbf{h}_v^{(L,K_L)}, v \in \mathcal{V}$. Lastly, a READOUT operation is applied to get the graph-level representation,

$$\mathbf{h}_G = \text{READOUT}(\{\mathbf{h}_v^{(0,K_0)}, ..., \mathbf{h}_v^{(L,K_L)} \mid v \in \mathcal{V}\}). \tag{4}$$

4 The GROVER Pre-training Framework

This section contains details of our pre-training architecture together with the well-designed self-supervision tasks. On a high level, the model is a Transformer-based neural network with tailored GNNs as the self-attention building blocks. The GNNs therein enable capturing structural information in the graph data and information flow on both the node and edge message passing paths. Furthermore, we introduce a dynamic message passing scheme in the tailored GNN, which is proved to boost the generalization performance of GROVER models.

4.1 Details of Model Architecture

GROVER consists of two modules: the node GNN transformer and edge GNN transformer. In order to ease the exposition, we will only explain details of the node GNN transformer (abbreviated as node GTransformer) in the sequel, and ignore the edge GNN transformer since it has a similar structure. Figure 1 demonstrates the overall architecture of node GTransformer. More details of GROVER are deferred to Appendix A.

GNN Transformer (GTransformer)**.** The key component of the node GTransformer is our proposed *graph multi-head attention* component, which is the attention blocks tailored to structural input data. A vanilla attention block, such as that in Equation (1), requires vectorized inputs. However, graph inputs are naturally structural data that are not vectorized. So we design a tailored GNNs (dyMPN, see the following sections for details) to extract vectors as queries, keys and values from nodes of the graph, then feed them into the attention block.

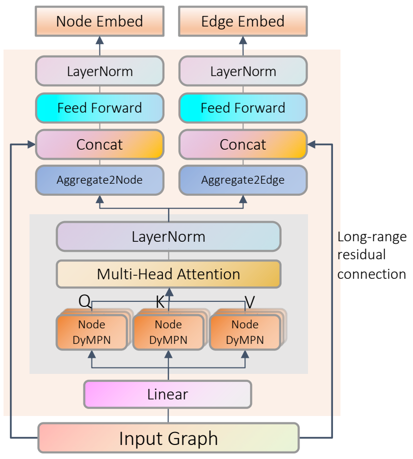

Figure 1: Overview of GTransformer.

This strategy is *simple* yet *powerful*, because it enables utilizing the highly expressive GNN models, to better model the structural information in molecular data. The high expressiveness of GTransformer can be attributed to its *bi-level* information extraction framework. It is well-known that the message passing process captures local structural information of the graph, therefore using the outputs of the GNN model as queries, keys and values would get the local subgraph structure involved, thus constituting the first level of information extraction. Meanwhile, the Transformer encoder can be viewed as a variant of the GAT [22, 54] on a fully connected graph constructed by $\mathcal{V}$. Hence, using Transformer encoder on top of these queries,

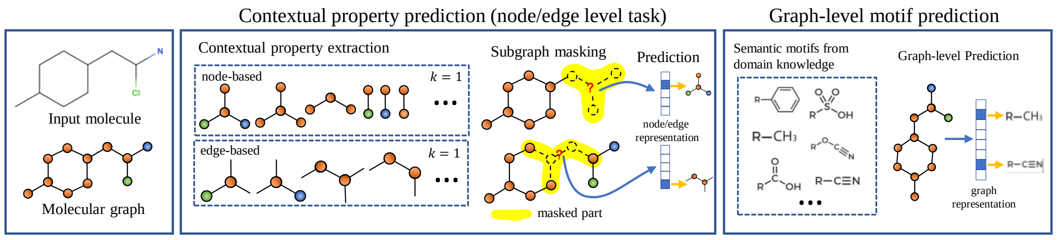

Figure 2: Overview of the designed self-supervised tasks of GROVER.

keys and values makes it possible to extract global relations between nodes, which enables the second level of information extraction. This bi-level information extraction strategy largely enhances the representational power of GROVER models.

Additionally, GTransformer applies a *single long-range residual connection* from the input feature to convey the initial node/edge feature information directly to the last layers of GTransformer instead of multiple short-range residual connections in the original Transformer architecture. Two benefits could be obtained from this single long-range residual connection: i) like ordinary residual connections, it improves the training process by alleviating the vanishing gradient problem [17], ii) compared to the various short-range residual connections in the Transformer encoder, our long-range residual connection can alleviate the *over-smoothing* [20, 34] problem in the message passing process.

Dynamic Message Passing Network (dyMPN**).** The general message passing process (see Equation (3)) has two hyperparameters: number of iterations/layers L and number of hops $K_l, l = 1, ..., L$ within each iteration. The number of hops is closely related to the size of the receptive field of the graph convolution operation, which would affect generalizability of the message passing model.

Given a fixed number of layers L, we find out that the pre-specified number of hops might not work well for different kinds of dataset. Instead of pre-specified K_l, we develop a randomized strategy for choosing the number of message passing hops during training process: at each epoch, we choose K_l from some random distribution for layer l. Two choices of randomization work well: i) $K_l \sim U(a, b)$, drawn from a uniform distribution; ii) K_l is drawn from a truncated normal distribution $\phi(\mu, \sigma, a, b)$, which is derived from that of a normally distributed random variable by bounding the random variable from both bellow and above. Specifically, let its support be $x \in [a, b]$, then the p.d.f. is $f(x) = \frac{\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}(\frac{x-\mu}{\sigma})^2\right]}{\sigma[\Phi(\frac{b-\mu}{\sigma})-\Phi(\frac{a-\mu}{\sigma})]}$, where $\Phi(x) = \frac{1}{2}(1 + \text{erf}(\frac{x}{\sqrt{2}}))$ is the cumulative distribution of a standard normal distribution.

The above randomized message passing scheme enables random receptive field for each node in graph convolution operation. We call the induced network Dynamic Message Passing networks (abbreviated as dyMPN). Extensive experimental verification demonstrates that dyMPN enjoys better generalization performance than vanilla message passing networks without the randomization strategy.

4.2 Self-supervised Task Construction for Pre-training

The success of the pre-training model crucially depends on the design of self-supervision tasks. Different from Hu et.al. [18], to avoid negative transfer on downstream tasks, we do not use the supervised labels in pre-training and propose new self-supervision tasks on both of these two levels: *contextual property prediction* and *graph-level motif prediction*, which are sketched in Figure 2.

Contextual Property Prediction. A good self-supervision task on the node level should satisfy the following properties: 1) The prediction target is reliable and easy to get; 2) The prediction target should reflect contextual information of the node/edge. Guided by these criteria, we present the tasks on both nodes and edges. They both try to predict the context-aware properties of the target node/edge within some local subgraph. What kinds of context-aware properties shall one use? We define recurrent statistical properties of local subgraph in the following two-step manner (let us take the node subgraph

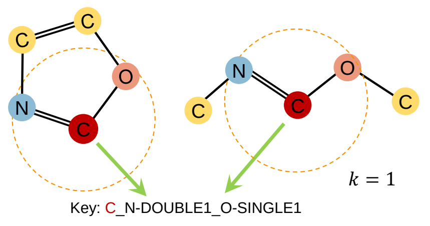

Figure 3: Illustration of contextual properties.

in Figure 3 as the example): i) Given a target node (e.g., the Carbon atom in red color), we extract its local subgraph as its k-hop neighboring nodes and edges. When k=1, it involves the Nitrogen atom, Oxygen atom, the double bond and single bond. ii) We extract statistical properties of this subgraph, specifically, we count the number of occurrence of (node, edge) pairs around the center node, which makes the term of `node-edge-counts`. Then we list all the node-edge counts terms in alphabetical order, which constitutes the final property: e.g., `C_N-DOUBLE1_O-SINGLE1` in the example. This step can be viewed as a clustering process: the subgraphs are clustered according to the extracted properties, one property corresponds to a cluster of subgraphs with the same statistical property.

With the context-aware property defined, the contextual property prediction task works as follows: given a molecular graph, after feeding it into the GROVER encoder, we obtain embeddings of its atoms and bonds. Suppose randomly choose the atom v and its embedding is $\mathbf{h}_v$. Instead of predicting the atom type of v, we would like $\mathbf{h}_v$ to encode some contextual information around node v. The way to achieve this target is to feed $\mathbf{h}_v$ into a *very* simple model (such as a fully connected layer), then use the output to predict the contextual properties of node v. This prediction is a multi-class prediction problem (one class corresponds to one contextual property).

Graph-level Motif Prediction. Graph-level self-supervision task also needs reliable and cheap labels. Motifs are recurrent sub-graphs among the input graph data, which are prevalent in molecular graph data. One important class of motifs in molecules are functional groups, which encodes the rich domain knowledge of molecules and can be easily detected by the professional software, such as RDKit [27]. Formally, the motif prediction task can be formulated as a *multi-label* classification problem, where each motif corresponds to one label. Suppose we are considering the presence of p motifs $\{\mathbf{m}_1, ..., \mathbf{m}_p\}$ in the molecular data. For one specific molecule (abstracted as a graph G), we use RDKit to detect whether each of the motif shows up in G, then use it as the target of the motif prediction task.

4.3 Fine-tuning for Downstream Tasks

After pre-training GROVER models on massive unlabelled data with the designed self-supervised tasks, one should obtain a high-quality molecular encoder which is able to output embeddings for both nodes and edges. These embeddings can be used for downstream tasks through the fine-tuning process. Various downstream tasks could benefit from the pre-trained GROVER models. They can be roughly divided into three categories: node level tasks, e.g., node classification; edge level tasks, e.g., link prediction; and graph level tasks, such as the property prediction for molecules. Take the graph level task for instance. Given node/edge embeddings output by the GROVER encoder, we can apply some READOUT function (Equation (4)) to get the graph embedding firstly, then use additional multiple layer perceptron (MLP) to predict the property of the molecular graph. One would use part of the supervised data to fine-tune both the encoder and additional parameters (READOUT and MLP). After several epochs of fine-tuning, one can expect a well-performed model for property prediction.

5 Experiments

Pre-training Data Collection. We collect 11 million (M) unlabelled molecules sampled from ZINC15 [48] and Chembl [11] datasets to pre-train GROVER. We randomly split 10% of unlabelled molecules as the validation sets for model selection.

Fine-tuning Tasks and Datasets. To thoroughly evaluate GROVER on downstream tasks, we conduct experiments on 11 benchmark datasets from the MoleculeNet [60] with various targets, such as quantum mechanics, physical chemistry, biophysics and physiology.[3] Details are deferred to Appendix B.1. In machine learning tasks, random splitting is a common process to split the dataset. However, for molecular property prediction, scaffold splitting [2] offers a more challenging yet realistic way of splitting. We adopt the scaffold splitting method with a ratio for train/validation/test as 8:1:1. For each dataset, as suggested by [60], we apply three independent runs on three random-seeded scaffold splitting and report the mean and standard deviations.

Baselines. We comprehensively evaluate GROVER against 10 popular baselines from MoleculeNet [60] and several state-of-the-arts (STOAs) approaches. Among them, TF_Roubust [40] is a DNN-based mulitask framework taking the molecular fingerprints as the input. GraphConv [24], Weave

[3]All datasets can be downloaded from `http://moleculenet.ai/datasets-1`

Table 1: The performance comparison. The numbers in brackets are the standard deviation. The methods in green are pre-trained methods.

Classification (Higher is better)						
Dataset # Molecules	BBBP 2039	SIDER 1427	ClinTox 1478	BACE 1513	Tox21 7831	ToxCast 8575
TF_Robust [40]	0.860$_{(0.087)}$	0.607$_{(0.033)}$	0.765$_{(0.085)}$	0.824$_{(0.022)}$	0.698$_{(0.012)}$	0.585$_{(0.031)}$
GraphConv [24]	0.877$_{(0.036)}$	0.593$_{(0.035)}$	0.845$_{(0.051)}$	0.854$_{(0.011)}$	0.772$_{(0.041)}$	0.650$_{(0.025)}$
Weave [23]	0.837$_{(0.065)}$	0.543$_{(0.034)}$	0.823$_{(0.023)}$	0.791$_{(0.008)}$	0.741$_{(0.044)}$	0.678$_{(0.024)}$
SchNet [45]	0.847$_{(0.024)}$	0.545$_{(0.038)}$	0.717$_{(0.042)}$	0.750$_{(0.033)}$	0.767$_{(0.025)}$	0.679$_{(0.021)}$
MPNN [13]	0.913$_{(0.041)}$	0.595$_{(0.030)}$	0.879$_{(0.054)}$	0.815$_{(0.044)}$	0.808$_{(0.024)}$	0.691$_{(0.013)}$
DMPNN [63]	0.919$_{(0.030)}$	0.632$_{(0.023)}$	0.897$_{(0.040)}$	0.852$_{(0.053)}$	0.826$_{(0.023)}$	0.718$_{(0.011)}$
MGCN [30]	0.850$_{(0.064)}$	0.552$_{(0.018)}$	0.634$_{(0.042)}$	0.734$_{(0.030)}$	0.707$_{(0.016)}$	0.663$_{(0.009)}$
AttentiveFP [61]	0.908$_{(0.050)}$	0.605$_{(0.060)}$	0.933$_{(0.020)}$	0.863$_{(0.015)}$	0.807$_{(0.020)}$	0.579$_{(0.001)}$
N-GRAM [29]	0.912$_{(0.013)}$	0.632$_{(0.005)}$	0.855$_{(0.037)}$	0.876$_{(0.035)}$	0.769$_{(0.027)}$	-[4]
HU. et.al[18]	0.915$_{(0.040)}$	0.614$_{(0.006)}$	0.762$_{(0.058)}$	0.851$_{(0.027)}$	0.811$_{(0.015)}$	0.714$_{(0.019)}$
GROVER$_{\text{base}}$	0.936$_{(0.008)}$	0.656$_{(0.006)}$	0.925$_{(0.013)}$	0.878$_{(0.016)}$	0.819$_{(0.020)}$	0.723$_{(0.010)}$
GROVER$_{\text{large}}$	**0.940**$_{(0.019)}$	**0.658**$_{(0.023)}$	**0.944**$_{(0.021)}$	**0.894**$_{(0.028)}$	**0.831**$_{(0.025)}$	**0.737**$_{(0.010)}$

Regression (Lower is better)					
Dataset # Molecules	FreeSolv 642	ESOL 1128	Lipo 4200	QM7 6830	QM8 21786
TF_Robust [40]	4.122$_{(0.085)}$	1.722$_{(0.038)}$	0.909$_{(0.060)}$	120.6$_{(9.6)}$	0.024$_{(0.001)}$
GraphConv [24]	2.900$_{(0.135)}$	1.068$_{(0.050)}$	0.712$_{(0.049)}$	118.9$_{(20.2)}$	0.021$_{(0.001)}$
Weave [23]	2.398$_{(0.250)}$	1.158$_{(0.055)}$	0.813$_{(0.042)}$	94.7$_{(2.7)}$	0.022$_{(0.001)}$
SchNet [45]	3.215$_{(0.755)}$	1.045$_{(0.064)}$	0.909$_{(0.098)}$	74.2$_{(6.0)}$	0.020$_{(0.002)}$
MPNN [13]	2.185$_{(0.952)}$	1.167$_{(0.430)}$	0.672$_{(0.051)}$	113.0$_{(17.2)}$	0.015$_{(0.002)}$
DMPNN [63]	2.177$_{(0.914)}$	0.980$_{(0.258)}$	0.653$_{(0.046)}$	105.8$_{(13.2)}$	0.0143$_{(0.002)}$
MGCN [30]	3.349$_{(0.097)}$	1.266$_{(0.147)}$	1.113$_{(0.041)}$	77.6$_{(4.7)}$	0.022$_{(0.002)}$
AttentiveFP [61]	2.030$_{(0.420)}$	0.853$_{(0.060)}$	0.650$_{(0.030)}$	126.7$_{(4.0)}$	0.0282$_{(0.001)}$
N-GRAM [29]	2.512$_{(0.190)}$	1.100$_{(0.160)}$	0.876$_{(0.033)}$	125.6$_{(1.5)}$	0.0320$_{(0.003)}$
GROVER$_{\text{base}}$	1.592$_{(0.072)}$	0.888$_{(0.116)}$	0.563$_{(0.030)}$	**72.5**$_{(5.9)}$	0.0172$_{(0.002)}$
GROVER$_{\text{large}}$	**1.544**$_{(0.397)}$	**0.831**$_{(0.120)}$	**0.560**$_{(0.035)}$	72.6$_{(3.8)}$	**0.0125**$_{(0.002)}$

[23] and SchNet [45] are three graph convolutional models. MPNN [13] and its variants DMPNN [63] and MGCN [30] are models considering the edge features during message passing. AttentiveFP [61] is an extension of the graph attention network. Specifically, to demonstrate the power of our self-supervised strategy, we also compare GROVER with two pre-trained models: N-Gram [29] and Hu et.al [18]. We only report classification results for [18] since the original implementation do not admit regression task without non-trivial modifications.

Experimental Configurations. We use Adam optimizer for both pre-train and fine-tuning. The Noam learning rate scheduler [9] is adopted to adjust the learning rate during training. Specific configurations are:

GROVER **Pre-training.** For the contextual property prediction task, we set the context radius $k = 1$ to extract the contextual property dictionary, and obtain 2518 and 2686 distinct node and edge contextual properties as the node and edge label, respectively. For each molecular graph, we randomly mask 15% of node and edge labels for prediction. For the graph-level motif prediction task, we use RDKit [27] to extract 85 functional groups as the motifs of molecules. We represent the label of motifs as the one-hot vector. To evaluate the effect of model size, we pre-train two GROVER models, GROVER$_{\text{base}}$ and GROVER$_{\text{large}}$ with different hidden sizes, while keeping all other hyper-parameters the same. Specifically, GROVER$_{\text{base}}$ contains $\sim$48M parameters and GROVER$_{\text{large}}$ contains $\sim$100M parameters. We use 250 Nvidia V100 GPUs to pre-train GROVER$_{\text{base}}$ and GROVER$_{\text{large}}$. Pre-training GROVER$_{\text{base}}$ and GROVER$_{\text{large}}$ took 2.5 days and 4 days respectively. For the models depicted in Section 5.2, we use 32 Nvidia V100 GPUs to pre-train the GROVER model and its variants.

Fine-tuning Procedure. We use the validation loss to select the best model. For each training process, we train models for 100 epochs. For hyper-parameters, we perform the random search on the validation set for each dataset and report the best results. More pre-training and fine-tuning details are deferred to Appendix C and Appendix D.

[4]The result is not presented since N-Gram on ToxCast is too time consuming to be finished in time.

5.1 Results on Downstream Tasks

Table 1 documents the overall results of all models on all datasets, where the cells in gray indicate the previous SOTAs, and the cells in blue indicates the best result achieved by GROVER. Table 1 offers the following observations: (1) GROVER models consistently achieve the best performance on all datasets with large margin on most of them. The overall relative improvement is 6.1% on all datasets (2.2% on classification tasks and 10.8% on regression tasks).[5]. This remarkable boosting validates the effectiveness of the pre-training model GROVER for molecular property prediction tasks. (2) Specifically, $\text{GROVER}_{\text{base}}$ outperforms the STOAs on 8/11 datasets, while $\text{GROVER}_{\text{large}}$ surpasses the STOAs on all datasets. This improvement can be attributed to the high expressive power of the large model, which can encode more information from the self-supervised tasks. (3) In the small dataset FreeSolv with only 642 labeled molecules, GROVER gains a 23.9% relative improvement over existing SOTAs. This confirms the strength of GROVER since it can significantly help with the tasks with very little label information.

5.2 Ablation Studies on Design Choices of the GROVER Framework

5.2.1 How Useful is the Self-supervised Pre-training?

To investigate the contribution of the self-supervision strategies, we compare the performances of pre-trained GROVER and GROVER without pre-training on classification datasets, both of which follow the same hyper-parameter setting. We report the comparison of classification task in Table 2, it is not supervising that the performance of GROVER becomes worse without pre-training. The self-supervised pre-training leads to a performance boost with an average AUC increase of 3.8% over the model without pre-training. This confirms that the self-supervised pre-training strategy can learn the implicit domain knowledge and enhance the prediction performance of downstream tasks. Notably, the datasets with fewer samples, such as SIDER, ClinTox and BACE gain a larger improvement through the self-supervised pre-training. It re-confirms the effectiveness of the self-supervised pre-training for the task with insufficient labeled molecules.

5.2.2 How Powerful is GTransformer Backbone?

To verify the expressive power of GTransformer, we implement GIN and MPNN based on our framework. We use a toy data set with 600K unlabelled molecules to pre-train GROVER with different backbones under the same training setting with nearly the same number of parameters (38M parameters). As shown in Figure 4, GROVER with GTransformer backbone outperforms GIN and MPNN in both training and validation, which again verifies the effectiveness of GTransformer.

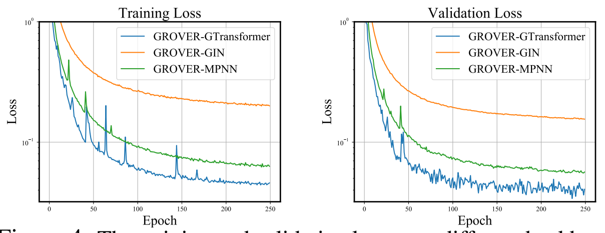

Figure 4: The training and validation losses on different backbones.

5.2.3 Effect of the Proposed dyMPN and GTransformer.

To justify the rationale behind the proposed GTransformer and dyMPN, we implement two variants: GROVER w/o dyMPN and GROVER w/o GTrans. GROVER w/o dyMPN fix the number of message passing hops K_l, while GROVER w/o GTrans replace the GTransformer with the original Transformer. We use the same toy data set to train GROVER w/o dyMPN and GROVER w/o GTrans under the same settings in Section 5.2.2. Figure 5 displays the curve of training and validation loss for three models. First, GROVER w/o GTrans is the worst one in both training and validation. It implies that trivially combining the GNN and Transformer can not enhance the expressive power of GNN. Second, dyMPN slightly harm the training loss by introducing randomness in the training process. However, the validation loss becomes better. Therefore, dyMPN brings a better generalization ability to GROVER by randomizing the receptive field for every message passing step. Overall, with new Transformer-style architecture and the dynamic message passing mechanism, GROVER enjoys high

[5]We use relative improvement [52] to provide the unified descriptions.

expressive power and can well capture the structural information in molecules, thus helping with various downstream molecular prediction tasks.

	GROVER	No Pretrain	Abs. Imp.
BBBP (2039)	**0.940**	0.911	+0.029
SIDER (1427)	**0.658**	0.624	+0.034
ClinTox (1478)	**0.944**	0.884	+0.060
BACE (1513)	**0.894**	0.858	+0.036
Tox21 (7831)	**0.831**	0.803	+0.028
ToxCast (8575)	**0.737**	0.721	+0.016
Average	**0.834**	0.803	+0.038

Table 2: Comparison between GROVER with and without pre-training.

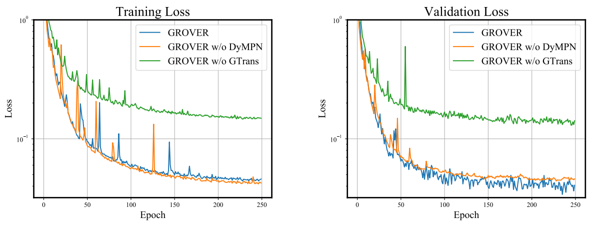

Figure 5: The training and validation loss of GROVER and its variants.

6 Conclusion and Future Works

We explore the potential of the large-scale pre-trained GNN models in this work. With well-designed self-supervised tasks and largely-expressive architecture, our model GROVER can learn rich implicit information from the enormous unlabelled graphs. More importantly, by fine-tuning on GROVER, we achieve huge improvements (more than 6% on average) over current STOAs on 11 challenging molecular property prediction benchmarks, which first verifies the power of self-supervised pre-trained approaches in the graph learning area.

Despite the successes, there is still room to improve GNN pre-training in the following aspects: **More self-supervised tasks.** Well designed self-supervision tasks are the key of success for GNN pre-training. Except for the tasks presented in this paper, other meaningful tasks would also boost the pre-training performance, such as distance-preserving tasks and tasks that getting 3D input information involved. **More downstream tasks.** It is desirable to explore a larger category of downstream tasks, such as node prediction and link prediction tasks on different kinds of graphs. Different categories of downstream tasks might prefer different pre-training strategies/self-supervision tasks, which is worthwhile to study in the future. **Wider and deeper models.** Larger models are capable of capturing richer semantic information for more complicated tasks, as verified by several studies in the NLP area. It is also interesting to employ even larger models and data than GROVER. However, one might need to alleviate potential problems when training super large models of GNN, such as gradient vanishing and oversmoothing.

Broader Impact

In this paper, we have developed a self-supervised pre-trained GNN model—GROVER to extract the useful implicit information from massive unlabelled molecules and the downstream tasks can largely benefit from this pre-trained GNN models. Below is the broader impact of our research:

- **For machine learning community:** This work demonstrates the success of pre-training approach on Graph Neural Networks. It is expected that our research will open up a new venue on an in-depth exploration of pre-trained GNNs for broader potential applications, such as social networks and knowledge graphs.

- **For the drug discovery community:** Researchers from drug discovery can benefit from GROVER from two aspects. First, GROVER has encoded rich structural information of molecules through the designing of self-supervision tasks. It can also produce feature vectors of atoms and molecule fingerprints, which can directly serve as inputs of downstream tasks. Second, GROVER is designed based on Graph Neural Networks and all the parameters are fully differentiable. So it is easy to fine-tune GROVER in conjunction with specific drug discovery tasks, in order to achieve better performance. We hope that GROVER can help with boosting the performance of various drug discovery applications, such as molecular property prediction and virtual screening.

Acknowledgements and Disclosure of Funding

This work is jointly supported by Tencent AI Lab Rhino-Bird Visiting Scholars Program (VS202006), China Postdoctoral Science Foundation (Grant No.2020M670337), and the National Natural Science Foundation of China (Grant No. 62006137). The GPU resources and distributed training optimization are supported by Tencent Jizhi Team. We would thank the anonymous reviewers for their valuable suggestions. Particularly, Yu Rong wants to thank his wife, Yunman Huang, for accepting his proposal for her hand in marriage.

References

[1] Tox21 challenge, 2017. https://tripod.nih.gov/tox21/challenge/.

[2] Guy W Bemis and Mark A Murcko. The properties of known drugs. 1. molecular frameworks. *Journal of medicinal chemistry*, 39(15):2887–2893, 1996.

[3] Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. Rumor detection on social media with bi-directional graph convolutional networks. In *AAAI 2020*, 2020.

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

[5] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug discovery today*, 23(6):1241–1250, 2018.

[6] Zhengdao Chen, Xiang Li, and Joan Bruna. Supervised community detection with line graph neural networks. *arXiv preprint arXiv:1705.08415*, 2017.

[7] Connor W Coley, Regina Barzilay, William H Green, Tommi S Jaakkola, and Klavs F Jensen. Convolutional embedding of attributed molecular graphs for physical property prediction. *Journal of chemical information and modeling*, 57(8):1757–1772, 2017.

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

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

[10] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[11] 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. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

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

[13] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272. JMLR. org, 2017.

[14] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[20] Wenbing Huang, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Tackling over-smoothing for general graph convolutional networks. *arXiv e-prints*, pages arXiv–2008, 2020.

[21] Stanisław Jastrzębski, Damian Leśniak, and Wojciech Marian Czarnecki. Learning to smile (s). *arXiv preprint arXiv:1602.06289*, 2016.

[22] Chaitanya Joshi. Transformers are graph neural networks, 2020. `https://graphdeeplearning.github.io/post/transformers-are-gnns/`.

[23] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[24] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[25] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations (ICLR)*, 2020.

[26] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079, 2015.

[27] Greg Landrum et al. Rdkit: Open-source cheminformatics. 2006.

[28] Jia Li, Yu Rong, Hong Cheng, Helen Meng, Wenbing Huang, and Junzhou Huang. Semi-supervised graph classification: A hierarchical graph perspective. In *The World Wide Web Conference*, pages 972–982. ACM, 2019.

[29] Shengchao Liu, Mehmet F Demirel, and Yingyu Liang. N-gram graph: Simple unsupervised representation for graphs, with applications to molecules. In *Advances in Neural Information Processing Systems*, pages 8464–8476, 2019.

[30] Chengqiang Lu, Qi Liu, Chao Wang, Zhenya Huang, Peize Lin, and Lixin He. Molecular property prediction: A multilevel quantum interactions modeling perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1052–1060, 2019.

[31] Jing Ma, Wei Gao, and Kam-Fai Wong. Detect rumors on twitter by promoting information campaigns with generative adversarial learning. In *The World Wide Web Conference*, pages 3049–3055, 2019.

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

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

[34] Kenta Oono and Taiji Suzuki. On asymptotic behaviors of graph cnns from dynamical systems perspective. *arXiv preprint arXiv:1905.10947*, 2019.

[35] Andrew Owens and Alexei A Efros. Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018.

[36] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pages 259–270, 2020.

[37] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[38] Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

[39] Raghunathan Ramakrishnan, Mia Hartmann, Enrico Tapavicza, and O Anatole Von Lilienfeld. Electronic spectra from tddft and machine learning in chemical space. *The Journal of chemical physics*, 143(8):084111, 2015.

[40] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

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

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

[43] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020.

[44] Seongok Ryu, Jaechang Lim, Seung Hwan Hong, and Woo Youn Kim. Deeply learning molecular structure-property relationships using attention-and gate-augmented graph convolutional network. *arXiv preprint arXiv:1805.10988*, 2018.

[45] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in neural information processing systems*, pages 991–1001, 2017.

[46] Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8(1):1–8, 2017.

[47] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*, 2018.

[48] Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

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

[50] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2019.

[51] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[52] Leo Törnqvist, Pentti Vartia, and Yrjö O Vartia. How should relative changes be measured? *The American Statistician*, 39(1):43–46, 1985.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

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

[55] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[56] Izhar Wallach, Michael Dzamba, and Abraham Heifets. Atomnet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*, 2015.

[57] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2017.

[58] Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. Smiles-bert: Large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 429–436, 2019.

[59] David Weininger, Arthur Weininger, and Joseph L Weininger. Smiles. 2. algorithm for generation of unique smiles notation. *Journal of chemical information and computer sciences*, 29(2):97–101, 1989.

[60] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[61] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. *Journal of medicinal chemistry*, 2019.

[62] Zheng Xu, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Seq2seq fingerprint: An unsupervised deep molecular embedding for drug discovery. In *BCB*, 2017.

[63] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[64] Liangzhen Zheng, Jingrong Fan, and Yuguang Mu. Onionnet: a multiple-layer intermolecular-contact-based convolutional neural network for protein–ligand binding affinity prediction. *ACS omega*, 4(14):15956–15965, 2019.

Appendix

A The Overall Architecture of GROVER Model

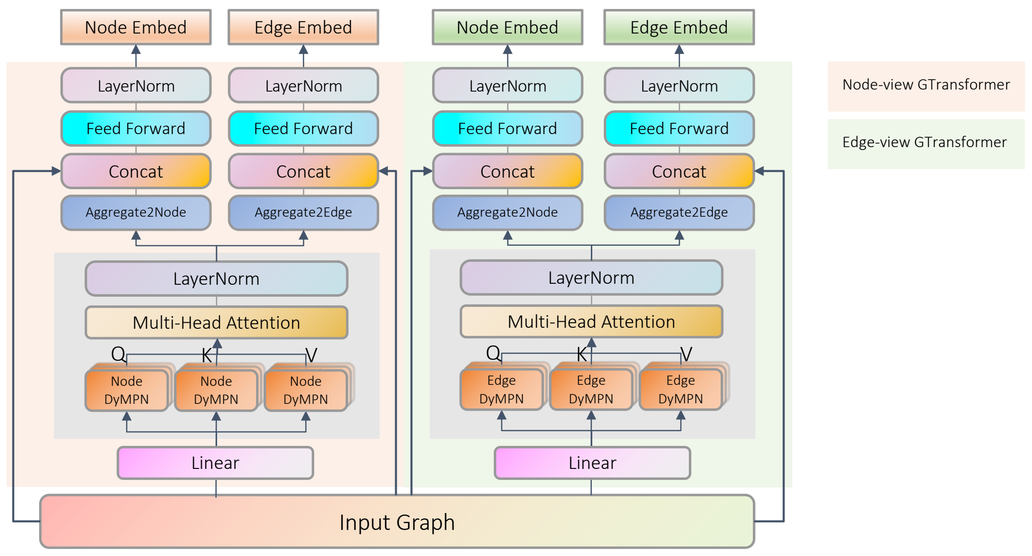

Figure 6: Overview of the whole GROVER architecture with both node-view GTransformer (in pink background) and edge-view GTransformer (in green background)

Figure 6 illustrates the complete architecture of GROVER models, which contains a node-view GTransformer (in pink background) and an edge-view GTransformer (in green background). Brief presentations of the node-view GTransformer have been introduced in the main text, and the edge-view GTransformer is in a similar structure. Here we elaborate more details of the GROVER model and its associated four sets of output embeddings.

As shown in Figure 6, node-view GTransformer contains node dyMPN, which maintains hidden states of nodes $\mathbf{h}_v, v \in \mathcal{V}$ and performs the message passing over nodes. Meanwhile, edge-view GTransformer contains edge dyMPN, that maintains hidden states of edges $\mathbf{h}_{vw}, \mathbf{h}_{wv}, (v, w) \in \mathcal{E}$ and conducts message passing over *edges*. The edge message passing is viewed as an ordinary message passing over the line graph of the original graph, where the line graph describes the neighboring of edges in the original graph and enables an appropriate way to define message passing over edges [6]. Note that edge hidden states have directions, i.e., $\mathbf{h}_{vw}$ is not identical to $\mathbf{h}_{wv}$ in general.

Then, after the multi-head attention, we denote the transformed node and edge hidden states by $\bar{\mathbf{h}}_v$ and $\bar{\mathbf{h}}_{vw}$, respectively.

Given the above setup, we can explain why GROVER will output four sets of embeddings in Figure 6. Let us focus on the information flow in the pink panel of Figure 6, first. Here the node hidden states $\bar{\mathbf{h}}_v$ encounter the two components, Aggregate2Node and Aggregate2Edge, which are used to aggregate the node hidden states to node messages and edge messages, respectively. Specifically, the Aggregate2Node and Aggregate2Edge components in node-view GTransformer is formulated as follows:

$$\mathbf{m}_v^{\text{node-embedding-from-node-states}} = \sum_{u \in \mathcal{N}_v} \bar{\mathbf{h}}_u \tag{5}$$

$$\mathbf{m}_{vw}^{\text{edge-embedding-from-node-states}} = \sum_{u \in \mathcal{N}_v \setminus w} \bar{\mathbf{h}}_u . \tag{6}$$

Then the node-view GTransformer transforms the node messages $\mathbf{m}_v^{\text{node-embedding-from-node-states}}$ and edge messages $\mathbf{m}_{vw}^{\text{edge-embedding-from-node-states}}$ through Pointwise Feed Forward layers [53] and Add&LayerNorm to produce the final node embeddings and edge embeddings, respectively.

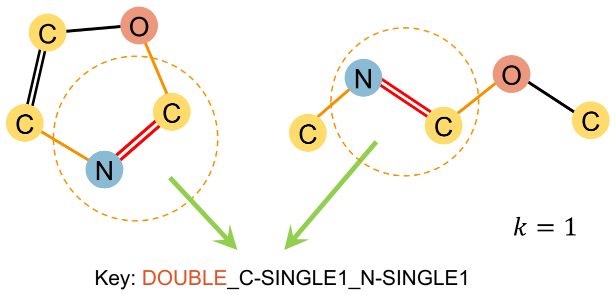

Figure 7: Examples of constructing contextual properties for edges

Similarly, for the information flow in the green panel, the edge hidden states $\bar{\mathbf{h}}_{vw}$ encounter the two components Aggregate2Node and Aggregate2Edge as well. Their operations are formulated as follows,

$$\mathbf{m}_v^{\text{node-embedding-from-edge-states}} = \sum_{u \in \mathcal{N}_v} \bar{\mathbf{h}}_{uv}, \tag{7}$$

$$\mathbf{m}_{vw}^{\text{edge-embedding-from-edge-states}} = \sum_{u \in \mathcal{N}_v \setminus w} \bar{\mathbf{h}}_{uv}. \tag{8}$$

Then, the edge-view GTransformer transforms the node messages and edge messages through Pointwise Feed Forward layers and Add&LayerNorm to produce the final node embeddings and edge embeddings, respectively.

In summary, the GROVER model outputs four sets of embeddings from two information flows. The node information flow (node GTransformer) maintains node hidden states and finally transform them into another node embeddings and edge embeddings, while the edge information flow (edge GTransformer) maintains edge hidden states and also transforms them into node and edge embeddings. The four sets of embeddings reflect structural information extracted from the two distinct views, and they are flexible to conduct downstream tasks, such as node-level prediction, edge-level prediction and graph-level prediction (via an extra READOUT component).

A.1 Fine-tuning Model for Molecular Property Prediction

As explained above, given a molecular graph G_i and the corresponding label $\boldsymbol{y}_i$, GROVER produces two node embeddings, $\mathbf{H}_{i,\text{node-view}}$ and $\mathbf{H}_{i,\text{edge-view}}$, from node-view GTransformer and edge-view GTransformer, respectively. We feed these two node embeddings into a shared self-attentive READOUT function to generate the graph-level embedding [28, 54]:

$$\begin{aligned} \mathbf{S} &= \operatorname{softmax}\left(\mathbf{W}_2 \tanh\left(\mathbf{W}_1 \mathbf{H}^\top\right)\right), \\ \boldsymbol{g} &= \operatorname{Flatten}(\mathbf{S}\mathbf{H}), \end{aligned} \tag{9}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d_{\text{attn\_hidden}} \times d_{\text{hidden\_size}}}$ and $\mathbf{W}_2 \in \mathbb{R}^{d_{\text{attn\_out}} \times d_{\text{attn\_hidden}}}$ are two weight matrix and $\boldsymbol{g}$ is the final graph embedding. After the READOUT, we employ two distinct MLPs to generate two predictions: $\boldsymbol{p}_{i,\text{node-view}}$ and $\boldsymbol{p}_{i,\text{edge-view}}$. Besides the supervised loss $\mathcal{L}(\boldsymbol{p}_{i,\text{node-view}}, \boldsymbol{y}_i) + \mathcal{L}(\boldsymbol{p}_{i,\text{edge-view}}, \boldsymbol{y}_i)$, the final loss function also includes a disagreement loss [28] $\mathcal{L}_{\text{diss}} = ||\boldsymbol{p}_{i,\text{node-view}} - \boldsymbol{p}_{i,\text{edge-view}}||_2$ to retrain the consensus of two predictions.

A.2 Constructing Contextual Properties for Edges

In Section 4.2 we describe an example of constructing contextual properties of nodes, here we present an instance of cooking edge contextual properties in order to complete the picture.

Similar to the process of node contextual property construction, we define recurrent statistical properties of local subgraph in a two-step manner. Let us take the graphs in Figure 7 for instance and consider the double chemical bond in red color in the left graph.

Step I: We extract its local subgraph as its k-hop neighboring nodes and edges. When k=1, it involves the Nitrogen atom, Carbon atom and the two single bonds. Step II: We extract statistical properties of this subgraph, specifically, we count the number of occurrence of (node, edge) pairs around the center edge, which makes the term of `node-edge-counts`. Then we list all the node-edge counts terms in alphabetical order, which makes the final property: e.g., `DOUBLE_C_SINGLE1_N-SINGLE1` in the example.

Note that there are two graphs and two double bonds in red color in Figure 7, since their subgraphs have the same statistical property, the resulted contextual properties of the two bonds would be the same. For a different point of view, this step can be viewed as a clustering process: the subgraphs are clustered according to the extracted properties, one property corresponds to a cluster of subgraphs with the same statistical property.

B Details about Experimental Setup

B.1 Dataset Description

Table 3: Dataset information

Type	Category	Dataset	# Tasks	# Compounds	Metric
Classification	Biophysics	BBBP	1	2039	ROC-AUC
	Physiology	SIDER	27	1427	ROC-AUC
		ClinTox	2	1478	ROC-AUC
		BACE	1	1513	ROC-AUC
		Tox21	12	7831	ROC-AUC
		ToxCast	617	8575	ROC-AUC
Regression	Physical chemistry	FreeSolv	1	642	RMSE
		ESOL	1	1128	RMSE
		Lipophilicity	1	4200	RMSE
	Quantum mechanics	QM7	1	6830	MAE
		QM8	12	21786	MAE

Table 3 summaries information of benchmark datasets, including task type, dataset size, and evaluation metrics. The details of each dataset are listed bellow [60]:

Molecular Classification Datasets.

- `BBBP` [32] involves records of whether a compound carries the permeability property of penetrating the blood-brain barrier.
- `SIDER` [26] records marketed drugs along with its adverse drug reactions, also known as the Side Effect Resource .
- `ClinTox` [12] compares drugs approved through FDA and drugs eliminated due to the toxicity during clinical trials.
- `BACE` [49] is collected for recording compounds which could act as the inhibitors of human β-secretase 1 (BACE-1) in the past few years.
- `Tox21` [1] is a public database measuring the toxicity of compounds, which has been used in the 2014 Tox21 Data Challenge.
- `ToxCast` [41] contains multiple toxicity labels over thousands of compounds by running high-throughput screening tests on thousands of chemicals.

Molecular Regression Datasets.

- `QM7` [4] is a subset of GDB-13, which records the computed atomization energies of stable and synthetically accessible organic molecules, such as HOMO/LUMO, atomization energy, etc. It contains various molecular structures such as triple bonds, cycles, amide, epoxy, etc .
- `QM8` [39] contains computer-generated quantum mechanical properties, e.g., electronic spectra and excited state energy of small molecules.
- `ESOL` is a small dataset documenting the solubility of compounds [8].

- `Lipophilicity` [11] is selected from the ChEMBL database, which is an important property that affects the molecular membrane permeability and solubility. The data is obtained via octanol/water distribution coefficient experiments .
- `FreeSolv` [33] is selected from the Free Solvation Database, which contains the hydration free energy of small molecules in water from both experiments and alchemical free energy calculations .

Dataset Splitting. We apply the scaffold splitting [2] for all tasks on all datasets. It splits the molecules with distinct two-dimensional structural frameworks into different subsets. It is a more challenging but practical setting since the test molecular can be structurally different from training set. Here we apply the scaffold splitting to construct the train/validation/test sets.

B.2 Feature Extraction Processes for Molecules

The feature extraction contains two parts: 1) Node / edge feature extraction. We use RDKit to extract the atom and bond features as the input of dyMPN. Table 4 and Tabel 5 show the atom and bond feature we used in GROVER. 2) Molecule-level feature extraction. Following the same protocol of [60, 63], we extract additional 200 molecule-level features by RDKit for each molecule and concatenate these features to the output of self-attentive READOUT, to go through MLP for the final prediction.

Table 4: Atom features.

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

Table 5: Bond features.

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

C Implementation and Pre-training Details

We use Pytorch to implement GROVER and horovod [47] for the distributed training. We use the Adam optimizer with learning rate 0.00015 and L2 weight decay for 10^{-7}. We train the model for 500 epochs. The learning rate warmup over the first two epochs and decreases exponentially from 0.00015 to 0.00001. We use PReLU [16] as the activation function and the dropout rate is 0.1 for all layers. Both GROVER$_{\text{base}}$ and GROVER$_{\text{large}}$ contain 4 heads. We set the iteration $L = 1$ and sample $K_l \sim \phi(\mu = 6, \sigma = 1, a = 3, b = 9)$ for the embedded dyMPN in GROVER. $\phi(\mu, \sigma, a, b)$ is a truncated normal distribution with a truncation range (a, b). The hidden size for GROVER$_{\text{base}}$ and GROVER$_{\text{base}}$ are 800 and 1200 respectively.

D Fine-tuning Details

For each task, we try 300 different hyper-parameter combinations via random search to find the best results. Table 6 demonstrates all the hyper-parameters of fine-tuning model. All fine-tuning tasks are run on a single P40 GPU.

Table 6: The fine-tuning hyper-parameters

hyper-parameter	Description	Range
batch_size	the input batch_size.	32
init_lr	initial learning rate ratio of Noam learning rate scheduler. The real initial learning rate is max_lr / init_lr.	10
max_lr	maximum learning rate of Noam learning rate scheduler.	$0.0001 \sim 0.001$
final_lr	final learning rate ratio of Noam learning rate scheduler. The real final learning rate is max_lr / final_lr.	$2 \sim 10$
dropout	dropout ratio.	0, 0.05, 0.1,0.2
attn_hidden	hidden size for the self-attentive readout.	128
attn_out	the number of output heads for the self-attentive readout.	4,8
dist_coff	coefficient of the disagreement loss	0.05, 0.1,0.15
bond_drop_rate	drop edge ratio [43]	0, 0.2,0.4,0.6
ffn_num_layer	The number of MLP layers.	2,3
ffn_hidden_size	The hidden size of MLP layers.	5,7,13

E Additional Experimental Results

E.1 Effect of Self-supervised Pre-training on Regression Tasks

Table 7 depicts the additional results of the comparison of the performance of pre-trained GROVER and GROVER without pre-training on regression tasks.

Table 7: Comparison between GROVER with and without pre-training on regression tasks

		GROVER	No Pre-training	Absolute Improvement
RMSE	FreeSolv	**1.544**	1.987	0.443
	ESOL	**0.831**	0.911	0.080
	Lipo	**0.560**	0.643	0.083
MAE	QM7	**72.600**	89.408	16.808
	QM8	**0.013**	0.017	0.004

E.2 GROVER Fine-tuning Tasks with Other Backbones

In order to verify the effectiveness of the proposed self-supervised tasks, we report the fine-tuning results by Hu et al. with and without pre-training in Table 8. As a comparison, we also involve the performance of GROVER with the backbone GIN and MPNN trained in Section 5.2. We find that without pre-training, our GROVER-GIN is consistent with Hu et al. on average, thus verifying the reliability of our implementations. However, after pre-training, GROVER-GIN achieves nearly 2% higher number than Hu et al., which supports the advantage of our proposed self-supervised loss.

Table 8: Comparison between different methods. The metric is AUC-ROC. The numbers in brackets are the standard deviation.

	Hu. et al.		GROVER-GIN		GROVER-MPNN	
	w pre-train	w/o pre-train	w pre-train	w/o pre-train	w pre-train	w/o pre-train
BBBP	$0.915_{(0.040)}$	$0.899_{(0.035)}$	$0.925_{(0.036)}$	$0.901_{(0.051)}$	$0.929_{(0.029)}$	$0.917_{(0.027)}$
SIDER	$0.614_{(0.006)}$	$0.615_{(0.007)}$	$0.648_{(0.015)}$	$0.627_{(0.016)}$	$0.650_{(0.003)}$	$0.637_{(0.030)}$
BACE	$0.851_{(0.027)}$	$0.837_{(0.028)}$	$0.862_{(0.020)}$	$0.823_{(0.050)}$	$0.872_{(0.031)}$	$0.852_{(0.034)}$
Average	0.793	0.784	0.812	0.784	0.817	0.802