**LiteGEM: Lite Geometry Enhanced Molecular Representation Learning for Quantum Property Prediction**

**Solution to KDD Cup 2021 PCQM4M-LSC Challenge**

Shanzhuo Zhang, Lihang Liu
Sheng Gao, Donglong He
PaddleHelix Team, Baidu Inc.
{zhangshanzhuo, liulihang}@baidu.com
{gaosheng06, hedonglong}@baidu.com

Weibin Li, Zhengjie Huang
Weiuye Su, Wenjin Wang
PGL Team, Baidu Inc.
{liweibin02, huangzhengjie}@baidu.com
{suweiyue, wangwenjin02}@baidu.com

**ABSTRACT**

In this report, we (SuperHelix team) present our solution to KDD Cup 2021-PCQM4M-LSC, a large-scale quantum chemistry dataset on predicting HOMO-LUMO gap of molecules. Our solution, **Lite Geometry Enhanced Molecular representation learning (LiteGEM)** achieves a mean absolute error (MAE) of 0.1204 on the test set with the help of deep graph neural networks and various self-supervised learning tasks. The code of the framework can be found in https://github.com/PaddlePaddle/PaddleHelix/tree/dev/competition/kddcup2021-PCQM4M-LSC/.

1 Introduction

Molecular property prediction has been widely considered as one of the most critical tasks in computational drug and materials discovery, since many methods rely on predicted molecular properties to evaluate, select and generate molecules. With the development of deep neural networks (DNNs), molecular representation learning exhibits a great advantage over feature engineering-based methods, which has attracted increasing research attention to tackle the molecular property prediction problem.

Inspired by our previous work ([Fang et al. 2021](#)), we propose using **Lite Geometry Enhanced Molecular representation learning (LiteGEM)** for the quantum property prediction: HOMO-LUMO gap ([Nakata and Shimazaki 2017](#)). We add the word “Lite” due to the fact that our original GEM model requires 3D geometry information as input feature, which is absent in this dataset. However, the self-supervised learning strategies proposed in GEM can still be incorporated into LiteGEM and boost the performance.

The report is organized as follows. We briefly introduce graph neural nets and message passing in Section 2. In Section 3, we present the main architecture of our model LiteGEM and its components. Experiment details such as choices of hyper-parameters and performance of ensemble can be found in Section 4.

2 Preliminaries

A molecule consists of atoms, and the neighboring atoms are connected by the chemical bonds, which can be naturally represented by a graph \( G = (V, E) \), where \( V \) is a node set and \( E \) is an edge set. An atom in the molecule is regarded as a node \( v \in V \) and a chemical bond in the molecule is regarded as an edge \((u, v) \in E\) connecting atoms \( u \) and \( v \).

Graph neural networks (GNNs) can be seen as message passing neural networks ([Gilmer et al. 2017](#)), which are useful for predicting molecular properties. Following the definitions of the previous GNNs ([Xu et al. 2019](#)), the features of a node \( v \) are represented by \( x_v \), and the features of an edge \((u, v)\) are represented by \( e_{uv} \). Taking node features, edge features and the graph structure as inputs, a GNN learns the representation vectors of the nodes and the

*Equal contribution.*
We introduce our solution in this section, including the GNN architectures, self-supervised learning strategies, and where $\| \cdot \|$ denotes concatenation of vectors, $\text{MLP}$ denotes a 2-layer Multi Layer Perceptron (MLP) with SiLU as the activation function and $\text{Linear}$ denotes the Linear layer. $\text{SoftMax}_\beta$ function is used as our aggregation, which is defined as:

$$\text{SoftMax}_\beta(\cdot) = \sum_{u \in \mathcal{N}(v)} \frac{\exp (\beta \mathbf{m}_{vu})}{\sum_{i \in \mathcal{N}(v)} \exp (\beta \mathbf{m}_{vi})},$$

where $\mathbf{m}_{vu} \in \mathbb{R}^D$ is the given message set $\{ \mathbf{m}_{vu} | u \in \mathcal{N}(v) \}$ and $\beta$ is the temperature controlling the smoothness of the distribution. Note that we also try to replace $\text{SoftMax}_\beta$ with a simple summation aggregation and observe no degradation on the performance, but we still keep it.

Overall, the LiteGEM consists of 11 LiteGEMConv layers with virtual node representations added at each layer. Node features $\mathbf{h}_v$ initialized as $\mathbf{x}_v$ are updated by LiteGEMConv layer-by-layer, and the output is denoted as $\mathbf{h}_v^{(K)}$. The graph level representation $\mathbf{h}_G$ is obtained via mean pooling over the node representations of all nodes and the final output as the prediction of HOMO-LUMO gap is produced by another MLP as:

$$y_G = \text{MLP}_G(\mathbf{h}_G).$$

Here $\text{MLP}_G$ is a 2- or 3-layer MLP with dropout, batch normalization and SiLU activation.
3.2 Auxiliary and Pre-training Tasks

Self-supervised learning methods have proven their effectiveness in computer vision and natural language tasks, when the number of labeled data is insufficient. To further enhance our model performance, we adopt both auxiliary tasks and pre-training tasks in a self-supervised fashion for graph neural networks. We follow our previous work (Fang et al. [2021]) and apply topology-level and geometry-level learning tasks as auxiliary and pre-training tasks.

3.2.1 Geometry-level: Bond Length & Bond Angle Prediction

According to the Hohenberg–Kohn theorems in DFT, the ground-state electron density determines all ground-state properties of a molecule, i.e., the ground-state conformation determines the HOMO-LUMO gap of a molecule. On the other hand, as proved in our previous work ChemRL-GEM (Fang et al. [2021]), by incorporating accurate 3D conformation in the QM9 dataset, we are able to achieve significant improvement in predicting various quantum properties of molecules. However, unlike ChemRL-GEM, due to the inference time limitation of this task, it is nearly impossible to generate accurate 3D conformation using DFT during inference.

To solve such dilemma, we adopt geometry-level self-supervised learning tasks of GEM only in the training procedure. More concretely, we utilize the bond length and bond angle prediction task of GEM as both the pre-training and auxiliary task for training LiteGEM. The bond lengths and the bond angles are the most important molecular geometrical parameters. The bond length is the distance between two joint atoms in a molecule, reflecting the bond strength between the atoms, while the bond angle is the angle connecting two consecutive bonds, including three atoms, describing the local spatial structure of a molecule.

3.2.2 Topology-level: Context Prediction

In context prediction, we use GNN to predict graph sub-structures (Mikolov et al. [2013], Rubenstein and Goodenough [1965]) using node-level representations. The intuition is that we want to enhance the ability of GNN to encode the neighborhood graph structure for each node. In order to represent the neighborhood of node \( u \) in a compact way, we design rules to map the neighborhood into a context string. Such string then contains all nodes and edges information that are at most 2 hops away from \( u \) in the graph. Specifically, the context string contains the following sub-strings:

- \( \text{str}(u) \);
- \( \text{str} (\text{sort}(\mathcal{N}(u))) \): the 1-hop neighboring nodes of \( u \);
- \( \text{str} (\text{sort} (\{ e_{uv} : v \in \mathcal{N}(u) \})) \): the 1-hop neighboring edges of \( u \);
- \( \text{str} (\text{sort} (\{ w : w \in \mathcal{N}(v), v \in \mathcal{N}(u) \})) \): the 2-hop neighboring nodes of \( u \);
- \( \text{str} (\text{sort} (\{ e_{vw} : w \in \mathcal{N}(v), v \in \mathcal{N}(u) \})) \): the 2-hop neighboring edges of \( u \);

where we use \( \mathcal{N}(v) \) to denote the neighboring nodes of node \( u \), the function \( \text{sort} \) is used to convert the set into a sorted array and the function \( \text{str} \) is used to convert an array into a compact string. The above 5 sub-strings can then can concatenated to construct the context string. Note that we use the atomic number as the information for the nodes and the bond type for the edges.

Upon obtaining the context string, we hash it into an integer between 0 and 5000 for each node. Then we construct a node-level multi-class classification task to predict these integers by add a Multi-layer Perceptron Layer (MLP) after the node representations. The MLP we used is composed of 2 linear layers with Batch Normalization and SiLU activation. The only hidden layer has size equal to half of the input dimension and the output layer has size of 5000. This fore-mentioned classification task is then used for both pre-training and auxiliary tasks.

3.3 Smoothing

To further improve the performance, inspired by APPNP algorithm (Klicpera et al. [2018]), we integrate a smoothing strategy after the last LiteGEMConv block. Our smoothing strategy utilizes the relationship between graph convolutional networks (GCN) and PageRank (Page et al. [1999]) to derive an improved propagation scheme based on personalized PageRank. It achieves linear computational complexity by approximating topic-sensitive PageRank via power iteration. The process is defined as:

\[
Z^{(0)} = M^L,
Z^{(k+1)} = (1 - \alpha) \hat{A}Z^{(k)} + \alpha M^L.
\]
Here, \( M^L \in \mathbb{R}^{N \times D} \), is the node representations \( h_v^{(K)} \) produced by the DeeperGCN. \( \hat{A} \) is the adjacency matrix with self-loops and normalization. Note that when \( \alpha = 0 \), the smoothing layer degenerates into an identity mapping.

### 3.4 Feature Selection

We extend the standard node and edge features provided by the OGB-PCQM4M dataset (Liu et al. [2021], Chen et al. [2019], Kearnes et al. [2016]). To be specific, we incorporate the atom type, chirality, degree, formal charge, number of hydrogen atoms connected, radical electron, hybridization, ring size, van der waals radius, valence of out shell, partial charge as atom features, and aromatic, if in ring, bond type, bond stereo type and conjugated as bond features.

### 4 Experiments

#### 4.1 Implementation Details

LiteGEM is implemented in PaddlePaddle\(^2\) and PGL\(^3\). The total number of parameters is about 74 million for a single model. Other hyper-parameters for training the model are listed in Table 1. The training time is about 100 minutes per epoch on an Nvidia Tesla V100. The model usually needs about 100 epochs to converge though we set the total number of epochs to 150. To fully make use of the validation set, we adopt a 2-fold cross-validation scheme. 90% of the original validation set is randomly split into 2 cross validation sets, and the last 10% is left to verify strategies of ensemble. We train 5 variants of LiteGEM and save 164 checkpoints for further model ensemble (see Table 2).

The whole training program is preceded by a pre-training stage, where the model is only trained with the self-supervised tasks for 10 epochs. After the pre-training, we combine the auxiliary tasks together with the HOMO-LUMO-gap task. To balance the trade-off between the auxiliary tasks and the HOMO-LUMO-gap task, we use a pre-chosen discount factor: the loss of auxiliary tasks is down-weighted by 1/2 at 20th epoch, and completely removed after 30th epoch. The initial weight for auxiliary tasks (\( w_{aux} \)) is shown in Table 1.

The 3D molecular conformation used in the bond length and bond angle tasks are generated by DFT using PySCF (Sun et al. [2020]). We mimic the DFT process used for calculating the original dataset (Nakata and Shimazaki [2017]) with various simplification to reduce resource consumption, and generate 3D molecular conformation for all the 3 million training SMILES. Due to the simplification, only 20% of DFT results are comparable with the original dataset and used for the auxiliary and pre-training tasks. The calculation for each molecule takes about 20 core minutes.

#### 4.2 Performance of Ensemble

Due to the limit of inference time, we need to control the size of ensembles. To achieve this, we first drop more than half the model checkpoints by the coefficients obtained from a Huber Regressor. Specifically, checkpoints with absolute coefficient less than 0.01 are deemed futile and dropped. All the 164 model checkpoints and the 73 checkpoints for ensemble are listed in Table 2. Original models are regressed on the 2 cross validation sets separately.

The predictions of remained checkpoints are then sorted by each compound and the largest and smallest 20% predictions are dropped. Remained predictions are regressed on another Huber Regressor. We use this regressor to produce final prediction on the test set.

Without ensemble, the best single model achieved an MAE of \( \sim 0.1235 \) on the last 10% validation set, and the ensemble further decreased the MAE to 0.1141. On the whole test set, the MAE of our final ensemble model is 0.1204.

---

\(^2\)https://github.com/paddlepaddle/paddle  
\(^3\)https://github.com/PaddlePaddle/PGL
Table 2: Number of model checkpoints before and after ensemble

<table>
<thead>
<tr>
<th>Auxiliary and Pre-training Tasks</th>
<th>ContextPred</th>
<th>BondLength</th>
<th>BondAngle</th>
<th>Smoothing</th>
<th># Ckpts</th>
<th># Ensemble Ckpts</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>36</td>
<td>9</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>40</td>
<td>15</td>
</tr>
<tr>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>24</td>
<td>15</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>-</td>
<td>24</td>
<td>14</td>
</tr>
<tr>
<td>Sum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>164</td>
<td>73</td>
</tr>
</tbody>
</table>

4.3 Inference Time

Taking raw SMILES as input, the total inference time for all the 377,423 test molecules is about 10.5 hours on an Nvidia Tesla P40. To be specific, it takes 0.5 hour in generating graph features, 10 hours in inference by all the 73 ensemble models (8 minutes each). The time consumed by the Huber Regressor for ensemble is negligible.

5 Conclusions and Future Work

In this challenge, our model LiteGEM achieves test MAE of 0.1204 on PCQM4M with the help of deep graph neural networks and self-supervised learning strategies. For future work, we will continue to study how to effectively incorporate knowledge from quantum mechanics. In the meantime, we sincerely thank the OGB team for their consistent hard work and effort in maintaining the contest and pushing the developments of graph neural networks.

References


