Highly Accurate Quantum Chemical Property Prediction with Uni-Mol+

Shuqi Lu[1], **Zhifeng Gao**[1], **Di He**[2], **Linfeng Zhang**[1], **Guolin Ke**[1]
[1]DP Technology
[2]Peking University
{lusq, gaozf}@dp.tech
dihe@pku.edu.cn, {zhanglf, kegl}@dp.tech

Abstract

Recent developments in deep learning have made remarkable progress in speeding up the prediction of quantum chemical (QC) properties by removing the need for expensive electronic structure calculations like density functional theory. However, previous methods that relied on 1D SMILES sequences or 2D molecular graphs failed to achieve high accuracy as QC properties are primarily dependent on the 3D equilibrium conformations optimized by electronic structure methods. In this paper, we propose a novel approach called `Uni-Mol+` to tackle this challenge. Firstly, given a 2D molecular graph, `Uni-Mol+` generates an initial 3D conformation from inexpensive methods such as RDKit. Then, the initial conformation is iteratively optimized to its equilibrium conformation, and the optimized conformation is further used to predict the QC properties. All these steps are automatically learned using Transformer models. We observed the quality of the optimized conformation is crucial for QC property prediction performance. To effectively optimize conformation, we introduce a two-track Transformer model backbone in `Uni-Mol+` and train it together with the QC property prediction task. We also design a novel training approach called linear trajectory injection to ensure proper supervision for the `Uni-Mol+` learning process. Our extensive benchmarking results demonstrate that the proposed `Uni-Mol+` significantly improves the accuracy of QC property prediction. We have made the code and model publicly available at `https://github.com/dptech-corp/Uni-Mol`.

1 Introduction

The design of new materials and molecules through computational methods heavily relies on the prediction of quantum chemical (QC) properties, such as free energy and HOMO-LUMO gap. High-throughput screening on a large database is the most straightforward approach to discovering molecules with desired properties [10]. However, this method can be cost-prohibitive due to the extensive computational requirements of electronic structure methods, which can take several hours to calculate properties on a single molecule. Therefore, finding ways to reduce the costs of calculating QC properties is a critical task. Hereafter, we take density functional theory (DFT) [11] as the default electronic structure method we consider.

Recent studies have demonstrated the potential of deep learning in accelerating QC property prediction [20, 32, 27]. This involves training a deep neural network model to predict the property using molecular inputs, thus replacing the heavy DFT calculations. Previous research has mainly utilized 1D SMILES [30, 25, 4] sequences or 2D molecular graphs [3, 21, 13, 23, 8, 32] as molecular inputs. However, predicting QC properties from 1D SMILES and 2D molecular graphs can be challenging due to their inherent discrepancy from the 3D equilibrium conformations optimized by DFT, which

Preprint. Under review.

are critical for QC property estimation. Therefore, establishing a direct relationship between the 1D/2D data and QC properties may be ineffective, despite their easy obtainability.

To address this challenge, in this paper, we propose a method called `Uni-Mol+`. Unlike previous approaches that directly predict QC properties from 1D/2D data, `Uni-Mol+` first generates raw 3D conformation from 1D/2D data using cheap methods, such as RDKit [16]. As the raw conformation is usually inaccurate, `Uni-Mol+` then optimizes it towards equilibrium conformation obtained from DFT and predicts QC properties with optimized conformation. In this framework, the quality of the optimized conformation is crucial to the performance of QC property prediction. Fortunately, we can use large-scale datasets (e.g., PCQM4MV2 benchmark) to build up millions of pairs of RDKit-generated conformation and high-quality DFT equilibrium conformation and learn the optimization process automatically, making the approach plausible. With a carefully designed model backbone and training tasks, `Uni-Mol+` shows superior performance in various benchmarks.

Model Backbone We have adopted a two-track Transformer backbone, similar to Uni-Mol [34]. This backbone consists of an atom representation track and a pair representation track. Besides, we propose several modifications to further enhance the model capacity. These include: i) An enhanced pair representation that is initialized with 3D spatial positional encoding and graph positional encoding. Subsequently, it is repeatedly updated via three operators: an outer product of atom representation, a triangular update operator, and a feed-forward network. ii) A 3D position prediction that uses attention to calculate position updates on the three axes. iii) Iterative optimization that improves the predicted 3D positions through iterations, towards DFT-generated conformations.

Training Strategy Utilizing DFT-optimized equilibrium conformations as the target, we present a method called *linear trajectory injection* (LTI) for efficient conformation optimization training. The proposed method involves selecting a random conformation from the trajectory between the cheap conformation and the DFT conformation, which is then taken as input to the model. Notably, since the actual trajectory is unknown in most cases, we use a pseudo trajectory, which assumes the movement between two conformations is a linear process. The LTI method provides several advantages, including a plethora of input conformations for a molecule, which serves as an efficient data augmentation strategy. Moreover, starting from an intermediate point in the pseudo trajectory simplifies training towards the DFT equilibrium conformation. Additionally, with iterative optimization, the model can progressively converge towards the target DFT equilibrium conformation along the trajectory.

Experimental Result Our proposed `Uni-Mol+` is evaluated on the widely-used PCQM4MV2 benchmark dataset [6], whose task is predicting the HOMO-LUMO gap based on molecular SMILES. Specifically, we leverage the DFT equilibrium conformations provided in the PCQM4MV2 training set, as well as the cheap conformations generated by RDKit, to train a `Uni-Mol+` model with LTI strategy. In the validation set of PCQM4MV2, by taking RDKit-generated conformations as inputs, our `Uni-Mol+` outperforms all previous state-of-the-art methods by a significant margin. The results provide clear evidence of the effectiveness of the proposed `Uni-Mol+`.

2 Related Work

Deep QC property prediction models with 1D/2D information Some prior works have solely utilized 1D information, such as SMILES sequences, to make predictions. These include Wang et al. [30], Ross et al. [25], and Gomez et al. [4]. However, to incorporate more information, a number of studies have employed 2D graphs or fingerprints derived from SMILES sequences. Gilmer et al. [3] utilized Graph Neural Network (GNN) to learn the molecular graph representation, while Graphormer [32] extended Transformer models to graph tasks through graph structural encodings. Numerous subsequent studies [3, 21, 13, 23, 8] have employed Transformer models on graph tasks and have made significant improvements and innovations by including self-attention mechanisms and relative positional encoding, thereby enhancing the capabilities of Transformer models in graph tasks.

Deep QC property prediction models with 3D information As 3D information is critical in predicting QC property, several recent studies have incorporated 3D information. Some works use 3D structure information to enhance the 2D representation during training. Stärk et al. [28] maximize the mutual information between 3D vectors and the 2D representations of a GNN to make them contain latent 3D information and Liu et al. [19] leverage the correspondence and consistency between 2D

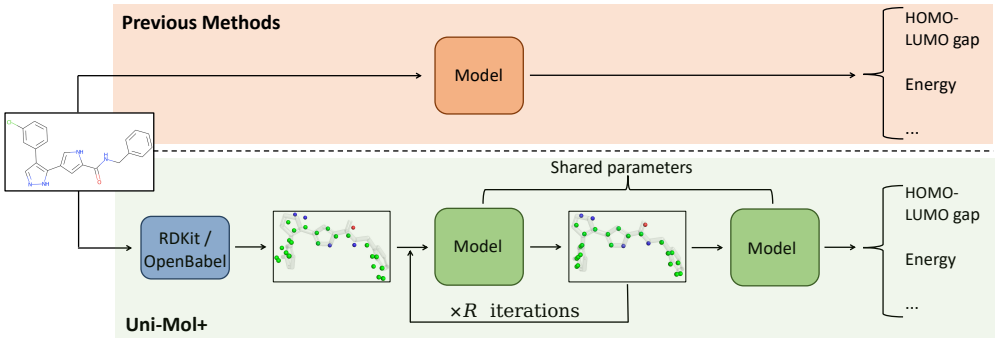

Figure 1: In contrast to prior methods that directly predict QC properties from 1D/2D data, `Uni-Mol+` employs a distinct approach. It firstly generates raw 3D conformation from 1D/2D data using cheap tools like RDKit, and subsequently optimizes it towards the equilibrium conformation obtained from DFT. Finally, it predicts QC properties using the optimized conformation.

topological structures and 3D geometric views to learn a 2D molecular graph encoder enhanced by richer and more discriminative 3D geometry. Zhu et al. propose a unified 2D and 3D pre-training scheme and make the 2D encoder and the 3D encoder promote each other. However, these works only implicitly embed 3D structural information into 2D representation, while during inference, only 2D information is used. Luo et al. [20] propose Transformer-M, an encoder capable of handling both 2D and 3D input formats. They are among the first to demonstrate that incorporating 3D information during training can enhance the performance of using 2D input alone in inference. However, the application of Transformer-M is restricted to taking 2D molecular graph or 3D DFT equilibrium conformation as input. Some works take 3D conformation as input [26, 15, 9, 29, 27, 32, 34] and consider the equivalence or invariance of rotation and translation in the model. For instance, Zhou et al. [34] introduce Uni-Mol, which utilizes the 3D conformation generated by RDKit as input and incorporates a 3D position recovery task for pretraining. Similarly, Shi et al. [27] propose Graphormer-3D, an extension of Graphormer [32] to a 3D Transformer model. Graphormer-3D takes the initial 3D conformation provided by the OC20 dataset [2] and predicts the energy at equilibrium. While this setup is similar to `Uni-Mol+`, the backbone of the model and the training strategy of the two models differ significantly.

3D conformation optimization Optimizing molecular conformations towards an equilibrium state is a crucial challenge in computational chemistry. The most popular method for this task is Density Functional Theory (DFT), which offers high accuracy, but at a substantial computational cost. To address this issue, several deep potential models have been proposed, such as DeePMD [33]. These models use neural networks to replace the costly potential calculations in DFT, thereby improving efficiency. However, deep potential models still require dozens or hundreds of steps to optimize the conformation iteratively based on the predicted potentials. In contrast, `Uni-Mol+` only requires a few rounds of optimization. Additionally, `Uni-Mol+` can end-to-end optimize the conformation, whereas deep potential models cannot. Although there are other works [5, 34] that also optimize RDKit-generated conformations towards DFT conformations, these works generally focus on benchmarking conformation rather than QC property. Furthermore, compared with `Uni-Mol+`, these works differ significantly in terms of model backbone and training strategy.

3 Method

For any molecule, `Uni-Mol+` first obtains an initial 3D conformation generated by cheap methods, such as template-based methods from RDKit and OpenBabel. It then optimizes the initial conformation iteratively towards the target conformation, i.e., the conformation generated by Density Functional Theory (DFT). Finally, using the optimized conformation as model input, `Uni-Mol+` learns to predict the QC properties. To achieve this objective, we introduce a new model backbone and a novel training strategy for optimizing conformation and predicting QC properties, which we will discuss in subsequent subsections.

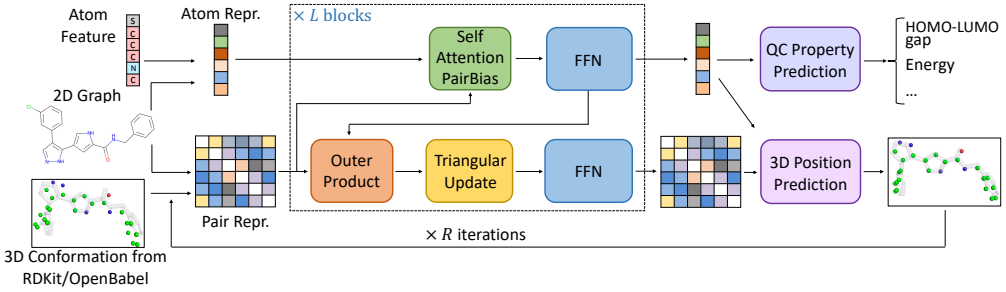

Figure 2: The `Uni-Mol+` model backbone consists of two tracks of representations - atom and pair, initialized by atom features and 2D graph/3D conformation respectively. These representations communicate with each other at every block. Besides, `Uni-Mol+` optimizes the predicted 3D position iteratively using the previous iteration's predicted conformations as input for the current iteration.

3.1 Model Backbone

We design a novel model backbone that can optimize conformation and predict QC property simultaneously, denoted as $(y, \hat{\boldsymbol{r}}) = f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{r}; \boldsymbol{\theta})$. This model uses atom features ($\boldsymbol{X} \in \mathbb{R}^{n \times d_f}$, where n is the number of atoms and d_f is atom feature dimension), edge features ($\boldsymbol{E} \in \mathbb{R}^{n \times n \times d_e}$, where d_e is the edge feature dimension), and 3D coordinates of atoms ($\boldsymbol{r} \in \mathbb{R}^{n \times 3}$), with learnable parameters $\boldsymbol{\theta}$, to predict a quantum property y and new 3D coordinates $\hat{\boldsymbol{r}} \in \mathbb{R}^{n \times 3}$. Like the Uni-Mol model [34], two tracks of representations are maintained: 1) Atom representation ($\boldsymbol{x} \in \mathbb{R}^{n \times d_x}$, where d_x is the dimension of the representation) and 2) Pair representation ($\boldsymbol{p} \in \mathbb{R}^{n \times n \times d_p}$, where d_p is the dimension of the representation). The model has L blocks, with $\boldsymbol{x}^{(l)}$ and $\boldsymbol{p}^{(l)}$ representing the l-th block's outputs. The overall architecture of `Uni-Mol+` is depicted in Fig. 2. Compared with Uni-Mol, there are several key differences, which will be outlined in the following paragraphs.

Positional Encoding Similar to previous works [32, 34], we use pair-wise encoding to encode the 3D spatial and 2D graph positional information. For 3D spatial information, we use the Gaussian kernel

$$s_{i,j}^k = \frac{1}{\sigma^k \sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\alpha_{i,j}||\boldsymbol{r}_i - \boldsymbol{r}_j|| - \mu^k + \beta_{i,j}}{\sigma^k}\right)^2\right), k = \{1, 2, ...K\}, \tag{1}$$

where $s_{i,j}^k$ is the k-th Gaussian kernel of the atom pair (i, j), K is the number of kernels. The input 3D coordinate of the i-th atom is represented by $\boldsymbol{r}_i \in \mathbb{R}^3$, and $\alpha_{i,j}$ and $\beta_{i,j}$ are learnable scalars indexed by the pair of atom types. μ^k and σ^k are predefined constants. Specifically, $\mu^k = w \times (k-1)/K$ and $\sigma^k = w/K$, where the width w is a hyper-parameter. Compared to Uni-Mol [34] and Transform-M [20], our approach includes fixed (non-learnable) μ^k and σ^k, which improves training stability. The 3D positional encoding is then obtained using a feed-forward network, represented by $\boldsymbol{\psi}_{i,j}^{\text{3D}} = \texttt{FFN}(\boldsymbol{s}_{i,j})$.

In addition to the 3D positional encodings, we also incorporate graph positional encodings similar to those used in Graphormer. This includes the shortest-path encoding, represented by $\boldsymbol{\psi}_{i,j}^{SP} = \texttt{Embedding}(\text{sp}_{ij})$ where sp_{ij} is the shortest path between atoms (i, j) in the molecular graph. Additionally, instead of the time-consuming multi-hop edge encoding method used in Graphormer, we utilize a more efficient one-hop bond encoding, denoted by $\boldsymbol{\psi}^{Bond} = \sum_{i=1}^{d_e} \texttt{Embedding}(\boldsymbol{E}_i)$, where $\boldsymbol{E}_i$ is the i-th edge feature.

Combined above, the positional encoding is denoted as $\boldsymbol{\psi} = \boldsymbol{\psi}^{\text{3D}} + \boldsymbol{\psi}^{\text{SP}} + \boldsymbol{\psi}^{\text{Bond}}$. And the pair representation $\boldsymbol{p}$ is initialized by $\boldsymbol{\psi}$, i.e., $\boldsymbol{p}^{(0)} = \boldsymbol{\psi}$.

Update of Atom Representation The atom representation $\boldsymbol{x}^{(0)}$ is initialized by the sum of embeddings of atom features $\boldsymbol{X}$ and the degree encodings from the graph structure, the same as Graphormer,

denoted as:

$$\boldsymbol{x}^{(0)} = \texttt{Embedding}(\boldsymbol{D}) + \sum_{i=1}^{d_f} \texttt{Embedding}(\boldsymbol{X}_i), \tag{2}$$

where $\boldsymbol{D}$ is the node degrees and $\boldsymbol{X}_i$ is the i-th atom feature. Then, at l-th block, we update the atom representation as follows:

$$\begin{aligned} \boldsymbol{x}^{(l)} &= \boldsymbol{x}^{(l-1)} + \texttt{SelfAttentionPairBias}\left(\boldsymbol{x}^{(l-1)}, \boldsymbol{p}^{(l-1)}\right), \\ \boldsymbol{x}^{(l)} &= \boldsymbol{x}^{(l)} + \texttt{FFN}\left(\boldsymbol{x}^{(l)}\right). \end{aligned} \tag{3}$$

The update process is very similar to the transformer layer, which contains a self-attention and an FFN layer, and the `SelfAttentionPairBias` can be denoted as:

$$\begin{aligned} \boldsymbol{Q}^{(l,h)} &= \boldsymbol{x}^{(l-1)}\boldsymbol{W}_Q^{(l,h)}; \quad \boldsymbol{K}^{(l,h)} = \boldsymbol{x}^{(l-1)}\boldsymbol{W}_K^{(l,h)}; \\ \boldsymbol{B}^{(l,h)} &= \boldsymbol{p}^{(l-1)}\boldsymbol{W}_B^{(l,h)}; \quad \boldsymbol{V}^{(l,h)} = \boldsymbol{x}^{(l-1)}\boldsymbol{W}_V^{(l,h)}; \\ \boldsymbol{A}^{(l,h)} &= \text{softmax}\left(\frac{\boldsymbol{Q}^{(l,h)}(\boldsymbol{K}^{(l,h)})^T}{\sqrt{d_h}} + \boldsymbol{B}^{(l,h)}\right)\boldsymbol{V}^{(l,h)}; \\ \boldsymbol{o} &= \text{concat}_h(\boldsymbol{A}^{(l,h)})\boldsymbol{W}_O^{(l)}, \end{aligned} \tag{4}$$

where d_h is the head dimension, $\boldsymbol{W}_Q^{(l,h)}, \boldsymbol{W}_K^{(l,h)}, \boldsymbol{W}_V^{(l,h)} \in \mathbb{R}^{d_x \times d_h}$, $\boldsymbol{W}_B^{(l,h)} \in \mathbb{R}^{d_p \times 1}$, $\boldsymbol{W}_O^{(l)} \in \mathbb{R}^{(d_h \times H) \times d_x}$, H is the number of heads, and $\boldsymbol{o}$ is the return result. Please note that the $\boldsymbol{o}$ is a temporary variable in the `SelfAttentionPairBias` function. For simplicity, layer normalization and dropout are omitted. In addition to the standard transformer layer, an attention bias term $\boldsymbol{B}^{(l,h)}$ has been incorporated within the self-attention. This bias, which is derived from the pair representation $\boldsymbol{p}^{(l-1)}$, is used to improve the pair-wise interactions between atoms.

Update of Pair Representation The pair representation $\boldsymbol{p}^0$ is initialized by the positional encoding ψ. The update process of pair representation begins with an outer product of $\boldsymbol{x}^{(l)}$, followed by a $\mathcal{O}(n^3)$ triangular multiplication, and is then concluded with an `FFN` layer. Formally, at l-th block, the update process is denoted as:

$$\begin{aligned} \boldsymbol{p}^{(l)} &= \boldsymbol{p}^{(l-1)} + \texttt{OuterProduct}(\boldsymbol{x}^{(l)}); \\ \boldsymbol{p}^{(l)} &= \boldsymbol{p}^{(l)} + \texttt{TriangularUpdate}(\boldsymbol{p}^{(l)}); \\ \boldsymbol{p}^{(l)} &= \boldsymbol{p}^{(l)} + \texttt{FFN}(\boldsymbol{p}^{(l)}). \end{aligned} \tag{5}$$

The `OuterProduct` is used for atom-to-pair communication, denoted as:

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{x}^{(l)}\boldsymbol{W}_{O1}^{(l)}; \\ \boldsymbol{b} &= \boldsymbol{x}^{(l)}\boldsymbol{W}_{O2}^{(l)}; \\ \boldsymbol{o}_{i,j} &= \text{flatten}(\boldsymbol{a}_i \otimes \boldsymbol{b}_j); \\ \boldsymbol{o} &= \boldsymbol{o}\boldsymbol{W}_{O3}^{(l)}, \end{aligned} \tag{6}$$

where $\boldsymbol{W}_{O1}^{(l)}, \boldsymbol{W}_{O2}^{(l)} \in \mathbb{R}^{d_x \times d_o}$, d_o is the hidden dimension of `OuterProduct`, and $\boldsymbol{W}_{O3}^{(l)} \in \mathbb{R}^{d_o^2 \times d_p}$, $\boldsymbol{o}$ is the return result. Please note that $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{o}$ are temporary variables in the `OuterProduct` function.

`TriangularUpdate` is used to further enhance pair representation, denoted as:

$$\begin{aligned} \boldsymbol{a} &= \text{sigmoid}\left(\boldsymbol{p}^{(l)}\boldsymbol{W}_{T1}^{(l)}\right) \odot \left(\boldsymbol{p}^{(l)}\boldsymbol{W}_{T2}^{(l)}\right); \\ \boldsymbol{b} &= \text{sigmoid}\left(\boldsymbol{p}^{(l)}\boldsymbol{W}_{T3}^{(l)}\right) \odot \left(\boldsymbol{p}^{(l)}\boldsymbol{W}_{T4}^{(l)}\right); \\ \boldsymbol{o}_{i,j} &= \sum_k \boldsymbol{a}_{i,k} \odot \boldsymbol{b}_{j,k} + \sum_k \boldsymbol{a}_{k,i} \odot \boldsymbol{b}_{k,j}; \\ \boldsymbol{o} &= \text{sigmoid}\left(\boldsymbol{p}^{(l)}\boldsymbol{W}_{T5}^{(l)}\right) \odot \left(\boldsymbol{o}\boldsymbol{W}_{T6}^{(l)}\right), \end{aligned} \tag{7}$$

where $\boldsymbol{W}_{T1}^{(l)}, \boldsymbol{W}_{T2}^{(l)}, \boldsymbol{W}_{T3}^{(l)}, \boldsymbol{W}_{T4}^{(l)} \in \mathbb{R}^{d_p \times d_t}$, $\boldsymbol{W}_{T5}^{(l)} \in \mathbb{R}^{d_p \times d_p}$, $\boldsymbol{W}_{T6}^{(l)} \in \mathbb{R}^{d_t \times d_p}$, and d_t is the hidden dimension of `TriangularUpdate`. Please note that $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{g}, \boldsymbol{o}$ and k are temporary variables in the `TriangularUpdate` function. The `TriangularUpdate` is inspired by the Evoformer in AlphaFold [12]. The difference is that AlphaFold uses two modules, "outgoing" ($\boldsymbol{o}_{i,j} = \sum_k \boldsymbol{a}_{i,k} \odot \boldsymbol{b}_{j,k}$) and "incoming" ($\boldsymbol{o}_{i,j} = \sum_k \boldsymbol{a}_{k,i} \odot \boldsymbol{b}_{k,j}$) respectively. In `Uni-Mol+`'s `TriangularUpdate`, we merge the two modules into one, to save the computational cost.

3D Position Prediction The ability to predict 3D position is necessary in `Uni-Mol+`. Whereas in Uni-Mol, an SE(3)-equivariant head is used, in `Uni-Mol+`, we implement an attention-based head, similar to the one proposed in Graphormer-3D [27]. Formally, the 3D position prediction could be denoted as

$$
\begin{aligned}
\boldsymbol{P}^{(h)} &= \text{softmax}\left(\frac{\boldsymbol{Q}^{(h)}\left(\boldsymbol{K}^{(h)}\right)^T}{d_h} + \boldsymbol{B}^{(h)}\right); \\
\boldsymbol{d}_{i,j,k}^{(h)} &= \boldsymbol{P}_{i,j}^{(h)} \odot (\boldsymbol{r}_i - \boldsymbol{r}_j)_k, \quad k \in \{1,2,3\}; \\
\boldsymbol{d}_k &= \text{concat}_h\left(\boldsymbol{d}_k^{(h)} \boldsymbol{V}^{(h)}\right); \\
\boldsymbol{\delta}_k &= \boldsymbol{d}_k \boldsymbol{W}_O^k; \\
\hat{\boldsymbol{r}} &= \boldsymbol{r} + \eta \boldsymbol{\delta},
\end{aligned}
\tag{8}
$$

where $(\boldsymbol{r}_i - \boldsymbol{r}_j) \in \mathbb{R}^3$ represents the delta position of pair (i, j), $\boldsymbol{P}^{(h)}$ is the attention probabilities of the h-th head, $\boldsymbol{W}_O^k \in \mathbb{R}^{d_x \times 1}$ is a projection matrix, $\boldsymbol{\delta}_k \in \mathbb{R}^{n \times 1}$ denotes the position update on the k-th axis, η is a scalar hyperparameter controlling the step-size of the update, and $\hat{\boldsymbol{r}} \in \mathbb{R}^{n \times 3}$ represents the predicted 3D position. To simplify the presentation, we have not included the projections to obtain $\boldsymbol{Q}^{(h)}, \boldsymbol{K}^{(h)}, \boldsymbol{V}^{(h)}, \boldsymbol{B}^{(h)}$. Please note that $\boldsymbol{P}$ and $\boldsymbol{d}$ are temporary variables in this function. The 3D position update takes into account the overall rotation of the system by using the difference in positions $(\boldsymbol{r}_i - \boldsymbol{r}_j)$. However, it does not have the property of SE(3)-equivariance like in Uni-Mol. Despite this, we found it performs better than the SE(3)-equivariant head in Uni-Mol. To mitigate this lack of SE(3)-equivariance, random global rotations are applied to $\boldsymbol{r}$ during the training process.

Iterative Optimization The optimization process in many conformation optimization applications, such as Molecular Dynamics, is iterative. This approach is also employed in the `Uni-Mol+`. The number of iterations, denoted as R, is a hyperparameter. To distinguish the position from different iterations, the initial 3D position is denoted as $\boldsymbol{r}^{(0)} = \boldsymbol{r}$, and at the i-th iteration, the model can be denoted as $(y, \boldsymbol{r}^{(i)}) = f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{r}^{(i-1)}; \boldsymbol{\theta})$. It is noteworthy that parameters $\boldsymbol{\theta}$ are shared across all iterations, and only $r^{(i)}$ is updated iteratively during the iterations. Moreover, please note that the Iterative Optimization in `Uni-Mol+` involves only a few rounds, such as 2 or 4, instead of dozens or hundreds of steps in Molecular Dynamics.

3.2 Model Training

In the field of DFT geometry optimization or Molecular Dynamics simulations, a conformation is usually optimized through a step-by-step process, resulting in a trajectory of conformations. However, saving such a trajectory can be expensive, and publicly available datasets often only provide the optimized equilibrium conformations generated by DFT. The provision of a trajectory for training purposes would greatly facilitate the learning of conformation optimization, as intermediate states can be used to guide the model's training.

To address this issue, we propose a novel approach *linear trajectory injection* (LTI), which generates a pseudo trajectory and samples a random conformation from it as the model input during training. This approach allows us to use a set of equilibrium conformations generated by DFT as a reference for training, which is much more efficient than using full trajectories. Specifically, we adopt the assumption that the trajectory from a cheap conformation $\boldsymbol{r}$ to a target DFT conformation $\bar{\boldsymbol{r}}$ is a linear process. To sample an intermediate conformation along this trajectory, we employ the following sampling method:

$$
\boldsymbol{r}^{(0)} = q\boldsymbol{r} + (1-q)\left(\bar{\boldsymbol{r}} + \boldsymbol{c}\right), \tag{9}
$$

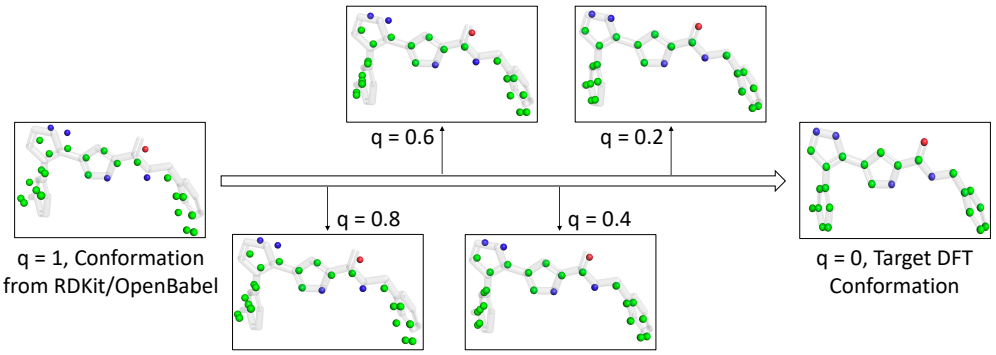

Figure 3: Illustration of the linear trajectory injection (LTI) method for conformational sampling. The leftmost cheap conformation is generated by RDKit/OpenBabel, while the rightmost conformation is the target DFT conformation. LTI assumes a linear trajectory from the left to the right and enables us to sample conformations by controlling the parameter q. To highlight the differences between the conformations, we include the target DFT conformation as a reference in translucent gray.

where scalar q is a sampling factor that ranges from 0 to 1. The intermediate conformation $\boldsymbol{r}^{(0)}$ is obtained by applying q to the cheap conformation and $(1-q)$ to the target DFT conformation, with the addition of Gaussian noise $\boldsymbol{c} \in \mathbb{R}^{n\times 3}$, which has a mean of 0 and standard deviation υ (a hyper-parameter). During inference, the sampling factor q is fixed to 1 because the target DFT conformation is unknown. During training, we can design a sampling strategy for q.

By utilizing randomly sampled q and noise $\boldsymbol{c}$, an extensive range of input conformations $\boldsymbol{r}^{(0)}$ can be obtained, which serves as an effective data augmentation. Furthermore, training from an intermediate point in the pseudo trajectory streamlines the process towards achieving the DFT equilibrium conformation. Additionally, through iterative optimization, the model can progressively approach the target DFT equilibrium conformation along the trajectory.

The model takes $\boldsymbol{r}^{(0)}$ as input and generates $\boldsymbol{r}^{(R)}$ after R iterations. The loss function for 3D conformation prediction is defined as follows:

$$\begin{aligned} \boldsymbol{d}_{ij}^{(R)} &= \left\| \boldsymbol{r}_i^{(R)} - \boldsymbol{r}_j^{(R)} \right\|_2, \quad \bar{\boldsymbol{d}}_{ij} = \left\| \bar{\boldsymbol{r}}_i - \bar{\boldsymbol{r}}_j \right\|_2, \\ \mathcal{L}_{3d} &= \gamma_{\text{coord}} \frac{1}{n} \left\| \boldsymbol{r}^{(R)} - \bar{\boldsymbol{r}} \right\|_1 + \gamma_{\text{dist}} \frac{1}{n^2} \left\| \boldsymbol{d}^{(R)} - \bar{\boldsymbol{d}} \right\|_1, \end{aligned} \tag{10}$$

where $\boldsymbol{d}_{ij}^{(R)}$ measures the distance between atoms i and j in the predicted conformation $\boldsymbol{r}^{(R)}$, $\bar{\boldsymbol{d}}_{ij}$ is the corresponding distance in the ground truth conformation $\bar{\boldsymbol{r}}$, and γ_{coord} and γ_{dist} are hyperparameters that control the weight of the coordinate loss and the distance loss, respectively. Notably, the loss is only calculated based on the last iteration, despite the fact that there are R iterations. The gradients are backpropagated through all iterations via $\boldsymbol{r}^{(i)}$.

Similar to Graphormer [32], a virtual atom (node), which connects to all atoms, is used to represent the whole molecule. The quantum property is predicted based on the atom representation of the virtual atom, and L1 loss is used for the training. For the conformation learning, we include an additional regularization loss, which limits the global movement of the whole molecule to zero.

4 Experiment

This section presents an empirical analysis of the performance of `Uni-Mol+`. The configuration of `Uni-Mol+` is described first, followed by the results obtained from the PCQM4MV2 benchmark. We are currently conducting experiments on additional datasets and performing ablation studies, which will be included in the forthcoming version of this study.

Table 1: Performance on PCQM4MV2.

Model	# param.	# layers	Valid MAE (↓)	Leaderboard MAE (↓)
MLP-Fingerprint [7]	16.1M	-	0.1735	0.1760
GCN [14]	2.0M	-	0.1379	0.1398
GIN [31]	3.8M	-	0.1195	0.1218
GINE-$_{\text{VN}}$ [1, 3, 20]	13.2M	-	0.1167	-
GCN-$_{\text{VN}}$ [14, 3]	4.9M	-	0.1153	0.1152
GIN-$_{\text{VN}}$ [31, 3]	6.7M	-	0.1083	0.1084
DeeperGCN-$_{\text{VN}}$ [17, 20]	25.5M	12	0.1021	-
GraphGPS$_{\text{SMALL}}$ [24]	6.2M	5	0.0938	-
TokenGT [13]	48.5M	12	0.0910	0.0919
GRPE$_{\text{BASE}}$ [23]	46.2M	12	0.0890	-
EGT [8]	89.3M	24	0.0869	0.0872
GRPE$_{\text{LARGE}}$ [8]	46.2M	18	0.0867	0.0876
Graphormer [32, 27]	47.1M	12	0.0864	-
GraphGPS$_{\text{BASE}}$ [24]	19.4M	10	0.0858	-
GraphGPS$_{\text{DEEP}}$ [24]	13.8M	16	0.0852	0.0862
GEM-2 [18]	32.1M	12	0.0793	0.0806
GPS++ [22]	44.3M	16	0.0778	0.0720 [1]
Transformer-M [20]	47.1M	12	0.0787	-
	69M	18	0.0772	0.0782
`Uni-Mol+`	27.7M	6	0.0751	-
	52.4M	12	0.0708	-
	77M	18	0.0701	0.0710

4.1 Model Configuration

Similar to both Graphormer [32] and Transformer-M [20], `Uni-Mol+` comprises 12 layers with an atom representation dimension of $d_x = 768$ and a pair representation dimension of $d_p = 256$. The model employs 128 Gaussian kernels, each with a width of $w = 9.0$ Å, and 48 attention heads in the `SelfAttentionPairBias` module. The hidden dimension of `FFN` in the atom representation track is set to 768, while that of the pair representation track is set to 256. Additionally, the hidden dimension in the `OuterProduct` is $d_o = 32$, and the hidden dimension in the `TriangularUpdate` is $d_t = 32$ as well. The number of iteration R is set to 2, and the position update step size is set to $\eta = 0.01$. For LTI training strategy, we specified a standard deviation of $\upsilon = 0.2$ for random noise and employed a particular sampling method for q. Specifically, q was set to 0.0 with probability 0.7, set to 1.0 with probability 0.2, and uniformly sampled from $[0.4, 0.6]$ with probability 0.1. For the weights of loss functions, we set γ_{coord} to 0.2, γ_{dist} to 1.0, and the weight for QC property loss to 1.0. With this setting, the number of parameters of `Uni-Mol+` is about 52.4M.

4.2 PCQM4MV2

Setting To evaluate the performance of `Uni-Mol+`, we tested it on the PCQM4Mv2 dataset from the OGB Large-Scale Challenge [6]. This dataset consists of 3.7 million training molecules, each with a SMILES representation, a DFT-generated 3D conformation, and a HOMO-LUMO gap label. We generated 8 conformations for each molecule using RDKit, at a per-molecule cost of approximately 0.01 seconds. During training, we randomly sampled 1 conformation as input $\boldsymbol{r}$ at each epoch, while during inference, we used the average HOMO-LUMO gap prediction based on 8 conformations. We used the AdamW optimizer with a learning rate of $2e$-4, a batch size of 1024, (β_1, β_2) set to $(0.9, 0.999)$, and gradient clipping set to 5.0 during training, which lasted for 1.5 million steps, with 150K warmup steps. We also utilized exponential moving average (EMA) with a decay rate of 0.999. The training process required around 5 days, powered by 8 NVIDIA A100 GPUs. Additionally, inference on the 147k test-dev set took approximately 7 minutes, utilizing 8 NVIDIA V100 GPUs.

[1] GPS++'s leaderboard submission consists of a 112-model ensemble and utilizes the validation data for training.

We incorporate previous submissions to the PCQM4MV2 leaderboard as baselines. In addition to the default 12-layer model, we evaluate the performance of Uni-Mol+ with two variants consisting of 6 and 18 layers, respectively, to investigate the impact of varying model parameter sizes.

Result We summarize the result in the Table 1. Our observations are as follows: 1) All three variants of Uni-Mol+ exhibit significant performance improvements over previous baselines. 2) Despite having considerably fewer model parameters, the 6-layer Uni-Mol+ still outperforms all previous baselines. 3) Further increasing the layers from 6 to 12 yields a substantial accuracy improvement, surpassing all baselines by a large margin. 4) The 18-layer Uni-Mol+ demonstrates the highest performance, outperforming all baselines by a significant margin. These outcomes are a clear testament to the efficacy of Uni-Mol+. 5) The performance of the single 18-layer Uni-Mol+ model on the leaderboard (test-dev set) is remarkable, especially considering that it has outperformed previous state-of-the-art methods without using an ensemble or other additional techniques. In comparison, the previous SOTA GPS++ relied on a 112-model ensemble and incorporated the validation set for training.

4.3 Discussion

Computational Cost The computational complexity of Uni-Mol+ is directly linked to the number of atoms n in the system. When the number of atoms is small, such as in small molecules, the runtime cost is acceptable. For example, in PCQM4MV2, the per-molecule cost is approximately 0.02 seconds when utilizing a single V100 GPU. However, for larger systems, we can decrease the cost in two ways: firstly, by removing the TriangularUpdate component, which is the most time-consuming aspect of Uni-Mol+; and secondly, by reducing self-attention and pair-representation from fully connected pairs $n \times n$ to top-k-nearest cutoff (or radius cut-off) pairs $n \times k$.

Initial Conformation In an ideal scenario, Uni-Mol+ would learn how DFT optimizes conformations, and it would be preferable to use the input conformation for DFT as the input for Uni-Mol+. Unfortunately, public datasets like QM9 and PCQM4MV2 do not provide initial conformations, so Uni-Mol+ relies on RDKit to generate them in these datasets. However, running DFT with these RDKit's generated conformations may not yield the same equilibrium conformations as in the datasets. If the actual initial DFT conformation were provided in these benchmarks, the performance of Uni-Mol+ would be further improved.

5 Conclusion

This paper presents Uni-Mol+, a new model to improving the accuracy of quantum chemical property prediction through deep learning. Our method optimizes conformations generated by cheap methods towards equilibrium ones generated by DFT, to predict quantum chemical properties with high accuracy. To facilitate effective learning of the conformation optimization process, we developed a new two-track Transformer backbone and linear trajectory injection training task. Our experimental results demonstrate the effectiveness of Uni-Mol+ and our belief that it has significant potential for future research in this field.

References

[1] Rémy Brossard, Oriel Frigo, and David Dehaene. Graph convolutions that can finally model local structure. *arXiv preprint arXiv:2011.15069*, 2020.

[2] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, et al. Open catalyst 2020 (oc20) dataset and community challenges. *Acs Catalysis*, 11(10):6059–6072, 2021.

[3] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[4] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel,

Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[5] Jiaqi Guan, Wesley Wei Qian, qiang liu, Wei-Ying Ma, Jianzhu Ma, and Jian Peng. Energy-inspired molecular conformation optimization. In *International Conference on Learning Representations*, 2022.

[6] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. OGB-LSC: A large-scale challenge for machine learning on graphs. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[7] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[8] Md Shamim Hussain, Mohammed J Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 655–665, 2022.

[9] Michael J Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. In *International Conference on Machine Learning*, pages 4533–4543. PMLR, 2021.

[10] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1):011002, 2013.

[11] Robert O Jones. Density functional theory: Its origins, rise to prominence, and future. *Reviews of modern physics*, 87(3):897, 2015.

[12] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[13] Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *arXiv preprint arXiv:2207.02505*, 2022.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs 2020. *arXiv preprint arXiv:2003.03123*, 2003.

[16] Greg Landrum et al. Rdkit: Open-source cheminformatics software. 2016.

[17] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

[18] Lihang Liu, Donglong He, Xiaomin Fang, Shanzhuo Zhang, Fan Wang, Jingzhou He, and Hua Wu. Gem-2: Next generation molecular property prediction network with many-body and full-range interaction modeling. *arXiv preprint arXiv:2208.05863*, 2022.

[19] Shengchao Liu, Hanchen Wang, Weiyang Liu, Joan Lasenby, Hongyu Guo, and Jian Tang. Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*, 2021.

[20] Shengjie Luo, Tianlang Chen, Yixian Xu, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. One transformer can understand both 2d & 3d molecular data. *arXiv preprint arXiv:2210.01765*, 2022.

[21] Shengjie Luo, Shanda Li, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. Your transformer may not be as powerful as you expect. *arXiv preprint arXiv:2205.13401*, 2022.

[22] Dominic Masters, Josef Dean, Kerstin Klaser, Zhiyi Li, Sam Maddrell-Mander, Adam Sanders, Hatem Helal, Deniz Beker, Ladislav Rampášek, and Dominique Beaini. Gps++: An optimised hybrid mpnn/transformer for molecular property prediction. *arXiv preprint arXiv:2212.02229*, 2022.

[23] Wonpyo Park, Woong-Gi Chang, Donggeon Lee, Juntae Kim, et al. Grpe: Relative positional encoding for graph transformer. In *ICLR2022 Machine Learning for Drug Discovery*, 2022.

[24] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *arXiv preprint arXiv:2205.12454*, 2022.

[25] Jerret Ross, Brian Belgodere, Vijil Chenthamarakshan, Inkit Padhi, Youssef Mroueh, and Payel Das. Large-scale chemical language representations capture molecular structure and properties. *Nature Machine Intelligence*, 4(12):1256–1264, 2022.

[26] 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. *Advances in neural information processing systems*, 30, 2017.

[27] Yu Shi, Shuxin Zheng, Guolin Ke, Yifei Shen, Jiacheng You, Jiyan He, Shengjie Luo, Chang Liu, Di He, and Tie-Yan Liu. Benchmarking graphormer on large-scale molecular modeling datasets, 2022.

[28] Hannes Stärk, Dominique Beaini, Gabriele Corso, Prudencio Tossou, Christian Dallago, Stephan Günnemann, and Pietro Liò. 3d infomax improves gnns for molecular property prediction. In *International Conference on Machine Learning*, pages 20479–20502. PMLR, 2022.

[29] Philipp Thölke and Gianni De Fabritiis. Equivariant transformers for neural network based molecular potentials. In *International Conference on Learning Representations*, 2022.

[30] 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.

[31] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[32] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[33] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and EJPRL Weinan. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical review letters*, 120(14):143001, 2018.

[34] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. In *The Eleventh International Conference on Learning Representations*, 2023.