# DeltaDock: A Unified Framework for Accurate, Efficient, and Physically Reliable Molecular Docking

**Jiaxian Yan**[1], **Zaixi Zhang**[1], **Jintao Zhu**[2], **Kai Zhang**[1], **Jianfeng Pei**[2], **Qi Liu**[1]*
[1]State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China
[2]Center for Quantitative Biology,
Academy for Advanced Interdisciplinary Studies, Peking University
`{jiaxianyan, zaixi, sa517494}@mail.ustc.edu.cn, zhujt@stu.pku.edu.cn,`
`jfpei@pku.edu.cn, qiliuql@ustc.edu.cn`

## Abstract

Molecular docking, a technique for predicting ligand binding poses, is crucial in structure-based drug design for understanding protein-ligand interactions. Recent advancements in docking methods, particularly those leveraging geometric deep learning (GDL), have demonstrated significant efficiency and accuracy advantages over traditional sampling methods. Despite these advancements, current methods are often tailored for specific docking settings, and limitations such as the neglect of protein side-chain structures, difficulties in handling large binding pockets, and challenges in predicting physically valid structures exist. To accommodate various docking settings and achieve accurate, efficient, and physically reliable docking, we propose a novel two-stage docking framework, DeltaDock, consisting of pocket prediction and site-specific docking. We innovatively reframe the pocket prediction task as a pocket-ligand alignment problem rather than direct prediction in the first stage. Then we follow a bi-level coarse-to-fine iterative refinement process to perform site-specific docking. Comprehensive experiments demonstrate the superior performance of DeltaDock. Notably, in the blind docking setting, DeltaDock achieves a 31% relative improvement over the docking success rate compared with the previous state-of-the-art GDL model DiffDock. With the consideration of physical validity, this improvement increases to about 300%.[†]

## 1 Introduction

Recent advancement in geometric deep learning (GDL) [1, 2, 3] presents an innovative and promising molecular docking paradigm to predict and understand the interactions between target proteins and drugs, which is of paramount importance for drug discovery [4, 5]. Unlike traditional docking methods that employ optimization algorithms to sample and identify best binding poses [6, 7], GDL methods interpret molecular docking as either a regression or generation task, eliminating the need for intensive candidate sampling [8, 9, 10]. Studies have demonstrated that GDL methods outperform their traditional counterparts, delivering enhancements in both the accuracy of binding pose predictions, as measured by the root-mean-square deviation (RMSD) metric, and the inference efficiency [11, 12].

According to whether a prior pocket is given, molecular docking can be divided into blind and site-specific docking [13]. Traditional sampling methods adeptly navigate both scenarios, primarily differing in the scope of the search space they explore. In contrast, GDL methods typically specialize in either one. For instance, EquiBind [8], and DiffDock [9] are designed for blind docking, neglecting the incorporation of binding pockets. Uni-Mol [14] and DiffBind-FR [15] concentrate on site-specific docking and only protein atomic level structure within a defined radius (usually 6-12 Å) of the

*Qi Liu is the corresponding author.
[†]All codes and data will be released on `https://github.com/jiaxianyan/DeltaDock`.

co-crystal is modeled. Despite some progress, these methods not only fail to handle two docking settings smoothly like traditional methods, but also confronted with certain limitations. For blind docking methods, they ignore the fine-grained protein side-chain structure. Regarding the site-specific docking methods, when dealing with pockets larger than the predetermined cutoff or when there is a requirement to model extensive pocket surrounding structures to account for long-range interactions, these methods significantly deteriorate in performance [16] and the demand for computational resources can escalate significantly, as evidenced in Appendix.A.2 and Appendix.A.3.

Besides these challenges, current GDL methods face additional limitations due to the lack of inductive biases, such as penalties for steric clashes or constraints on ligand mobility, leading to the generation of unrealistic docking poses. Buttenschoen et al. [16] proposed the PoseBusters test suit to verify and highlight these problems. In addition to the RMSD between predicted and ground-truth poses, the test suite incorporates 18 checks, encompassing chemical validity and consistency, intramolecular validity, and intermolecular validity. According to the test suite, the previously highest-performing method, DiffDock, achieves a success rate of only 14%. This is significantly lower than the 38% success rate achieved when chemical validity is not taken into account.

To resolve these problems, we propose **DeltaDock**, a unified GDL framework for accurate, efficient, and physically valid docking. DeltaDock is a two-stage framework consisting of a pocket prediction stage and a site-specific docking stage. With **"Delta"**, we mean that the optimal poses are predicted by iteratively refining the input structures in the second docking stage. The first pocket prediction stage is specialized for blind docking, where a binding pocket is identified from a set of candidates through a novel contrastive pocket-ligand alignment module CPLA. Then in the second stage, within the pockets predefined or selected by CPLA, binding structures are predicted in a bi-level coarse-to-fine iterative refinement module Bi-EGMN. This module prioritizes the residue-level structure covered by a large outer box (Fig.4) for pose positioning and coarse structure prediction. And the atom-level structure, within a relatively small radius from the coarse structure, is characterized for more refined predictions. In particular, the module incorporates (i) a GPU-accelerated pose sampling algorithm generating high-quality initial structure, (ii) a training objective imposing penalties for steric clashes and constraints on ligand mobility, and (iii) a rapid post-processing step composing torsional alignment and energy minimization for structure correction.

To accommodate two different docking settings, DeltaDock is specially designed as a two-stage framework rather than an end-to-end framework. Particularly, the pocket-ligand alignment module is inspired by the observation shown in Fig.5. Existing pocket prediction methods generally achieve a recall rate of just 70%-80%. However, when combining all possible pockets predicted by multiple methods, this recall rate reaches nearly 95%. According to this result, we shift the focus from designing increasingly powerful pocket prediction models to developing strategies for the effective selection of a candidate pocket from an ensemble of predicted pockets. The pocket prediction task is thus reframed as a pocket-ligand alignment problem innovatively. Regarding the site-specific docking stage, the key idea is to accurately predict reliable poses. Based on the proposed bi-level iterative refinement model, several components presented above are introduced additionally. Among them, the pose sampling algorithm is adopted for structure initialization, as previous works on structure prediction [17] have demonstrated the importance of a good initial structure. Other two components, namely the physics-informed training object and the fast structure correction step, are leveraged to ensure physical validity.

To demonstrate the effectiveness of DeltaDock, we performed comprehensive experiments to evaluate its predictive accuracy, efficiency, generalizability, and ability to predict physically valid binding poses. The experimental outcomes indicate that DeltaDock consistently surpasses the baseline methods in both blind docking and site-specific docking settings while maintaining remarkable computational efficiency. Notably, in the blind docking setting, DeltaDock exceeded the performance of the previous SOTA GDL method, DiffDock, by 30.8% in terms of the docking success rate, and it required only approximately 3.0 seconds per protein-ligand pair. With the consideration of physical validity, this improvement increases to approximately 300% on the PoseBusters benchmark.

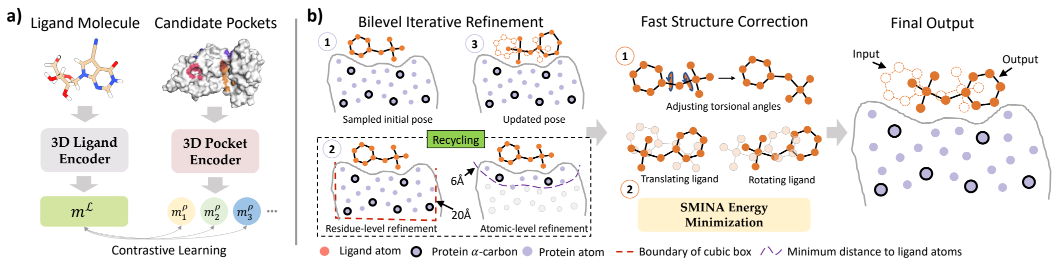

Figure 1: The overview of DeltaDock's two modules. (a) The pocket-ligand alignment module CPLA. Contrastive learning is adopted to maximize the correspondence between target pocket and ligand embeddings for training. During inference, the pocket with the highest similarity of the ligand is selected. (b) The bi-level iterative refinement module Bi-EGMN. Initialized with a high-quality sampled pose, the module first performs a coarse-to-fine iterative refinement. This process generates progressively refined ligand poses utilizing a recycling strategy. To guarantee the physical plausibility of the predicted poses, a two-step fast structure correction is subsequently applied. This correction involves torsion angle alignment followed by energy minimization based on the SMINA.

## 2 Related Work

### 2.1 Sampling-based Docking

Traditional docking methods, epitomized by the likes of VINA [18] and SMINA [19], operate on a "sampling-and-scoring" paradigm to identify the best binding pose. Optimization algorithms such as BFGS [20] are used to sample optimal poses within the defined search space on CPUs. This process, which involves a significant number of steps and multiple copies, is rather computationally intensive. Recent studies have attempted to speed up the sampling process using GPUs. Notable examples are Vina-GPU [21], Uni-Dock [22], and DSDP [23], which use more copies and shorter search steps to fully leverage the parallel computational power of GPUs. This approach has demonstrated substantial efficacy, achieving a speed increase of an order of magnitude compared to prior CPU-based methods.

### 2.2 Geometric Deep Learning-based Docking

GDL introduces a new paradigm in molecular docking, where the sampling process is bypassed by interpreting molecular docking as either a regression task or a generation task [8, 9]. However, recent researches have highlighted limitations of current GDL methods, such as neglect of protein side-chain structures [15], difficulties in handling large binding pockets, and challenges in predicting physically valid structures [16]. Compared with physically reliable sampling-based methods, especially recent developed GPU-accelerated methods, the existing limitations hinder the practical application of GDL methods. To address these concerns, in this work, we propose DeltaDock to overcome these problems and accomplish efficient, accurate, and physical reliable docking.

### 2.3 Binding Pocket Prediction

As the foundation of structure-based drug design, binding pocket prediction has attracted expansive attention. A variety of methods have been developed for this task, encompassing traditional computational methods, such as Fpocket [24], machine learning (ML) methods, such as P2Rank [25], and GDL methods, such as PUResNet [26]. These methods generally adopt ligand-free approaches and focus on predicting all potential binding sites within individual proteins. Recent blind docking methods, DSDP and FABind, apply pocket prediction for target ligands to reduce the docking search space, which is of great help to fast and accurate blind docking. In this study, our proposed model, DeltaDock, also prioritizes defining a pocket for blind docking. However, instead of improving model architecture for pocket prediction like previous methods, DeltaDock reframe the pocket prediction task as a pocket-ligand alignment problem and employ contrastive learning to select a candidate pocket from the combined pockets set.

## 3 DeltaDock Framework

### 3.1 Preliminaries

**Notations.** In this work, the separate structures of a protein $\mathcal{P}$ and a ligand $\mathcal{L}$ are used as inputs (Fig. 1). Both molecules are initially encoded as graphs, and we denote a molecule graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the node set and edge set respectively. Each node $v_i \in \mathcal{V}$ is associated with a coordinate $x_i$ and a feature vector $h_i$. Each edge $(i, j) \in \mathcal{E}$ is associated with an edge feature vector $e_{ij}$. For the ligand $\mathcal{L}$ and ligand graph $\mathcal{G}^{\mathcal{L}}$, $v_i^{\mathcal{L}}$ represents the $i$-th atom in the ligand and $x_i^{\mathcal{L}}$ corresponds to the atom's coordinate. For the protein $\mathcal{P}$, the situation is more complex, and two graphs based on the two structural levels of the protein are constructed. One is the protein atomic graph $\mathcal{G}^{\mathcal{P}}$, and the other is the protein residue graph $\mathcal{G}^{\mathcal{P}*}$. $\mathcal{G}^{\mathcal{P}}$ contains protein atomic-level information similar to ligand graph $\mathcal{G}^{\mathcal{L}}$, while $\mathcal{G}^{\mathcal{P}*}$ contains protein residue-level information and overlooks the side-chain structure information. In $\mathcal{G}^{\mathcal{P}*}$, $v_i^{\mathcal{P}*}$ represents the $i$-th residue in the protein and $x_i^{\mathcal{P}*}$ corresponds to the $C_\alpha$ coordinate of this residue. Details of the graph construction can be found in Appendix.A.6.

**Overview.** Our goal is to train a model $f$ that excels in both site-specific docking and blind docking scenarios of rigid molecular docking, wherein the protein structure is fixed and only the ligand's flexibility is considered.

*As depicted in Fig. 1, DeltaDock comprises two modules: a pocket-ligand alignment module* ***CPLA*** *responsible for selecting binding pocket from a pocket candidate set, and a bi-level iterative refinement module* ***Bi-EGMN*** *dedicated to executing site-specific docking given the binding pockets. This design allows DeltaDock to handle both blind docking and site-specific docking seamlessly. In the subsequent part of this section, we will elaborate on these two modules respectively.*

### 3.2 Contrastive Pocket-ligand Alignment

CPLA treats the pocket prediction task as a pocket-ligand alignment problem. We employ a list of well-established ligand-free pocket prediction methods to generate candidate pocket sets, and then map these pockets and the target ligand into the same embedding space. The correct pocket embedding is expected to have higher similarity with ligand embedding than other pockets.

#### 3.2.1 Data Preprocessing

The initial step of this module involves using RDKit [27] to generate a 3D conformer of the input ligand, as depicted in Fig. 1. Binding site prediction models including P2Rank and DSDP are adopted to extract druggable binding sites, and the binding sites predicted by these different methods are combined to form a set of candidate binding sites, denoted as $S = \{\varsigma_1, \varsigma_2, ...\}$, where $\varsigma_i$ represents the geometric center of $i$-th binding site. For CPLA, the protein pocket $\rho_i$ is defined as the residues within 15.0 Å to $\varsigma_i$.

#### 3.2.2 Ligand and Pocket Encoders

To map the ligand and pockets into the embedding space, the ligand encoder Attentive-FP (AFP) [28] and protein encoder Geometric Vector Perceptron (GVP) [29] are employed. These encoders first extract informative ligand node and protein node representations, and the feature extraction process can be formally expressed as:

$$H^{\mathcal{L}} = AFP(\mathcal{G}^{\mathcal{L}}),\ H^{\mathcal{P}*} = GVP(\mathcal{G}^{\mathcal{P}*}), \tag{1}$$

where $H^{\mathcal{L}}$ is the ligand embedding matrix of shape $|\mathcal{V}^{\mathcal{L}}| \times d$ and $H^{\mathcal{P}*}$ is the protein residue embedding matrix of shape $|\mathcal{V}^{\mathcal{P}*}| \times d$. The ligand representations $m^{\mathcal{L}}$ and pocket representations $m_i^{\rho}$ are then obtained by pooling ligand nodes embedding and pocket nodes embedding:

$$m^{\mathcal{L}} = Sum(H^{\mathcal{L}}, \mathcal{V}^{\mathcal{L}}),\ m_i^{\rho} = Sum(H^{\mathcal{P}*}, \mathcal{V}_i^{\rho}), \tag{2}$$

where $\mathcal{V}_i^{\rho}$ is the protein node set of $i$-th pocket $\rho$, and the pooling operation is sum pooling. For the pocket encoder, we input the entire protein residue graph $\mathcal{G}^{\mathcal{P}*}$ rather than just the protein pocket residue graph, to incorporate global protein information into the pocket representation.

### 3.2.3 Contrastive Embdding Alignment

With ligand representation $m^{\mathcal{L}}$ and pocket representation $m_i^{\rho}$ in hand, we calculate the cosine similarity score:

$$s_i = \frac{m^{\mathcal{L}} \cdot m_i^{\rho}}{\|m^{\mathcal{L}}\|_2 \cdot \|m_i^{\rho}\|_2}. \tag{3}$$

For the candidate pockets $S = \{\varsigma_1, \varsigma_2, ...\}$, the similarity score $s_+$ between the target pocket and the ligand is expected to be higher than others. Thus, we propose the contrastive learning objective:

$$L = -\frac{1}{N} \cdot log \frac{exp(s_+/\tau)}{\sum_i exp(s_i/\tau)}, \tag{4}$$

where $\tau$ is the temperature paramter. For blind docking, the pocket with the highest similarity score with the ligand is selected for the next docking step.

## 3.3 Bi-level Iterative Refinement

With a binding site $\varsigma$ predefined by the user or selected by CPLA, we design the bi-level iterative refinement module Bi-EGMN to predict binding pose within this pocket (Fig. 1).

### 3.3.1 Inital Structure Sampling

For an iterative refinement module, an initial structure is needed as a starting point. Previous work on molecular 3D conformer generation [17] demonstrates the importance of a good initial structure. Therefore, Bi-EGMN adopts a rapid GPU-accelerated sampling method proposed by Huang et al. [23] to sample a high-quality initial $\mathcal{X}^{\mathcal{L}}$. In this work, the search steps number and the search copy number are set to 40 and 384, respectively. Details about the search box setting can be found in the Appendix.B.3.1.

### 3.3.2 Structure Refinement

With input initial structure $\mathcal{X}^{\mathcal{L}}$, we iteratively update it to improve its accuracy. As discussed in Sec.1, the modeling of an entire binding pocket structure is crucial for the success of the process. Current methods either ignore the atom-level structure or model the full-atom pocket structure directly. The latter approach can significantly elevate the computational resource demand, particularly when dealing with large pockets. To overcome these challenges and maintain high docking accuracy and efficiency, we propose a bi-level strategy in this work. In the following sections, we first present the details of the bi-level strategy. Subsequently, we discuss the Bi-EGMN layer, which is used to perform refinement, as depicted in Fig. 1.

**Bi-level strategy.** The first refinement level is the residue level, where the protein residues within a 40.0 Å cubic region centered at the geometric centers of ligands are considered as pocket $\rho$. Previous work demonstrates such a range is large enough to cover the binding pocket [30]. In this context, as the full-atom structure of proteins is not considered, the pocket residue graph $\mathcal{G}^{\rho *}$ is adapted. The second level is the atomic level, where we set the ligand structures refined through $T$ rounds of residue level refinement as the reference structure. In this level, protein atoms within a 6.0 Å radius of the ligand atoms are considered to construct pocket atomic graph $\mathcal{G}^{\rho}$ for modeling the fine-grained interaction. The ligand coordinates $X^{a,\mathcal{L}}$ output by the last layer of atomic level refinement correspond to the final predicted structure $\hat{\mathcal{X}}^{\mathcal{L}}$.

**Bi-EGMN Layer**. The bi-level E(3)-equivariant graph matching network (Bi-EGMN) layer is the model designed to calculate the protein-ligand interaction and refine the structures. More specifically, this layer adheres to the message-passing paradigm [31] and consists of four functions: intra-message function, inter-message function, aggregate function, and update function.

The intra-message function works to extract messages $m_{i,j}$ and $\hat{m}_{i,j}$ between a node $i$ and its neighbor nodes $j$ from the same molecule graph. $m_{i,j}$ is later used for the updating of node features and $\hat{m}_{i,j}$ for the updating node coordinates. $\forall (i,j) \in \mathcal{E}_{\mathcal{P}} \cup \mathcal{E}_{\mathcal{L}}$, this function can be formally written as :

$$d_{i,j}^{(l)} = ||x_i^{(l)} - x_j^{(l)}||,\ m_{i,j} = \varphi_m(h_i^{(l)}, h_j^{(l)}, d_{i,j}^{(l)},),\ \hat{m}_{i,j} = (x_i^{(l)} - x_j^{(l)}) \cdot \varphi_{\hat{m}}(m_{i,j}), \tag{5}$$

where $d_{i,j}^{(l)}$ is the relative distance between node $i$ and node $j$, and $\varphi$ is a MLP.

The inter-message function works to extract messages $\mu_{i,j}$ and $\hat{\mu}_{i,j}$ between a node $i$ and its neighbor nodes $j$ from the other molecule graphs. Formally, $\forall i \in \mathcal{V}_{\mathcal{P}}, j \in \mathcal{V}_{\mathcal{L}}$ *or* $i \in \mathcal{V}_{\mathcal{L}}, j \in \mathcal{V}_{\mathcal{P}}$:

$$\mu_{i,j} = \varphi_{\mu}(h_i^{(l)}, h_j^{(l)}, d_{i,j}^{(l)}),\ \hat{\mu}_{i,j} = (x_i^{(l)} - x_j^{(l)}) \cdot \varphi_{\hat{\mu}}(\mu_{i,j}). \tag{6}$$

After extracting inter-message and intra-message, the aggregation function aggregates the neighbor messages of the node $i$. $\forall i \in \mathcal{V}_{\mathcal{P}} \cup \mathcal{V}_{\mathcal{L}}$:

$$m_i = \sum_{j \in \mathcal{N}(i)} m_{i,j},\ \hat{m}_i = \sum_{j \in \mathcal{N}(i)} \frac{1}{d_{i,j}^{(l)} + 1} \cdot \hat{m}_{i,j}, \tag{7}$$

$$\mu_i = \sum_{j \in \mathcal{N}_*^{(l)}(i)} \varphi(\mu_{i,j}) \cdot \mu_{i,j},\ \hat{\mu}_i = \sum_{j \in \mathcal{N}_*^{(l)}(i)} \frac{1}{d_{i,j}^{(l)} + 1} \cdot \hat{\mu}_{i,j}, \tag{8}$$

where $\mathcal{N}(i)$ is the neighbor of node $i$ in the same graph, and $\mathcal{N}_*^{(l)}(i)$ is the set of nodes associated with node $i$ in the other graph.

Finally, the update function updates the position and features of each node:

$$x_i^{(l+1)} = \eta x_i^{(0)} + (1-\eta) x_i^{(l)} + \hat{m}_i + \hat{\mu}_i,\ \forall i \in \mathcal{V}_{\mathcal{L}}, \tag{9}$$

$$h_i^{(l+1)} = (1-\beta) \cdot h_i^{(l)} + \beta \cdot \varphi(h_i^{(l)}, m_i, \mu_i, h_i^{(0)}),\ \forall i \in \mathcal{V}_{\mathcal{P}} \cup \mathcal{V}_{\mathcal{L}}, \tag{10}$$

where $\beta$ and $\eta$ are feature skip connection weight and coordinates skip connection weight, respectively. Through such a message-passing paradigm, our Bi-EGMN layers make to update coordinates iteratively.

#### 3.3.3 Fast Structure Correction

Lastly, as Bi-EGMN updates structures by modifying the coordinates rather than the torsional angles, as is done in methods like DiffDock [9] and other sampling-based methods, it is crucial to ensure the plausibility of bond lengths and bond angles of the updated structure $\hat{\mathcal{X}}^{\mathcal{L}}$. Therefore, fast structure correction steps, torsion alignment, and SMINA-based energy minimization are designed.

**Torsion Alignment.** We employ a rapid torsion alignment for the updated structure. The target of this alignment is to align the input structure $\mathcal{X}^{\mathcal{L}}$ with the updated structures $\hat{\mathcal{X}}^{\mathcal{L}}$ by rotating its torsional bonds. Formally, let $(b_i, c_i)$ denote a $i$-th rotatable bond, where $b_i$ and $c_i$ are the starting and ending atoms of the bond, respectively. We randomly select a neighboring atom $a_i$ of $b_i$ and a neighbor atom $d_i$ of $c_i$ to calculate the dihedral angle $\hat{\delta}_i = \angle(a_i b_i c_i, b_i c_i d_i)$ based on updated structure coordinates $\hat{\mathcal{X}}^{\mathcal{L}}$. Subsequently, we rotate the rotatable bond $(b_i, c_i)$ of input structures to match its dihedral angle $\delta_i$ the same as $\hat{\delta}_i$. This simple operation can be implemented efficiently using RDKit. After all rotatable bonds have been rotated, we align the rotated input structure to the updated structures to obtain the torsionally aligned structure $\hat{\mathcal{X}}^{\mathcal{L}\mathcal{T}}$. This process ensures the plausibility of bond lengths and bond angles in the torsionally aligned structure $\hat{\mathcal{X}}^{\mathcal{L}\mathcal{T}}$.

**Energy Minimization.** To further enhance the reliability of DeltaDock, we implement an energy minimization on the torsionally aligned structure $\hat{\mathcal{X}}^{\mathcal{L}\mathcal{T}}$, when an inter-molecular steric clash between the protein and ligand is detected. This energy minimization is conducted using SMINA [19], as it is a highly efficient tool for this process compared with specialized energy minimization tool OpenMM [32] (details see Appendix.A.5). The output structure of this process is $\hat{\mathcal{X}}^{\mathcal{L}'}$.

### 3.4 Training and Inference

#### 3.4.1 CPLA

The training object $L$ is a contrastive object defined before (Eq. 4). For a protein and its candidate pockets set $S = \{\varsigma_1, \varsigma_2, ...\}$, the positive pair is the target pocket-ligand pair and the negative pairs are other pocket-ligand pairs. The pocket-ligand pairs across different proteins are not used. When training, we calculate the minimum center distance ($DCC_{min}$) between all candidate pockets and the ligand. If $DCC_{min} \leq 5.0$ Å, we add the ligand center into $S$ to assert the existence of positive pairs for every protein (details see Appendix.B.3.1).

Table 1: Blind docking performance on the PDBbind dataset. All methods take RDKit-generated ligand structures and holo protein structures as input, trying to predict bound complex structures. DeltaDock-SC refers to the model variant that generates structures without implementing fast structure correction. DeltaDock-Random refers to the model variant that generates structures without high-quality initial poses. The best results are **bold**, and the second best results are underlined.

| Method | Time average | Time Split (363) | | | | Timesplit Unseen (142) | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | RMSD % below | | Centroid % below | | RMSD % below | | Centroid % below | |
| | Seconds | 2.0Å | 5.0Å | 2.0Å | 5.0Å | 2.0Å | 5.0Å | 2.0Å | 5.0Å |
| QVINA-W | 49* | 20.9 | 40.2 | 41.0 | 54.6 | 15.3 | 31.9 | 35.4 | 47.9 |
| GNINA | 393 | 21.2 | 37.1 | 36.0 | 52.0 | 13.9 | 27.8 | 25.7 | 39.5 |
| VINA | 119* | 10.3 | 36.2 | 32.3 | 55.2 | 7.8 | 25.5 | 24.1 | 41.8 |
| SMINA | 146* | 13.5 | 33.9 | 38.0 | 55.9 | 9.0 | 25.7 | 29.9 | 41.7 |
| GLIDE | 1405* | 21.8 | 33.6 | 36.1 | 48.7 | 19.6 | 28.7 | 29.4 | 40.6 |
| DSDP | 1.22 | 40.2 | 59.0 | 59.5 | 78.2 | 37.3 | 54.9 | 55.6 | 71.8 |
| EquiBind | **0.03** | 5.5 | 39.1 | 40.0 | 67.5 | 0.7 | 18.8 | 16.7 | 43.8 |
| TANKBind | 0.87 | 17.6 | 57.8 | 55.0 | 77.8 | 3.5 | 43.7 | 40.9 | 70.8 |
| DiffDock | 80 | 36.0 | 61.7 | 62.9 | 80.2 | 17.2 | 42.3 | 43.3 | 62.6 |
| FABind | 0.12 | 33.1 | 64.2 | 60.8 | 80.2 | 19.4 | 60.4 | 57.6 | 75.7 |
| FABind+ | 6.4 | 43.8 | **73.3** | 59.1 | **86.2** | 34.7 | **63.2** | 57.6 | 75.7 |
| **DeltaDock-SC** | 2.58 | **47.9** | 68.0 | **70.0** | 83.2 | **40.8** | 60.6 | **65.5** | **78.9** |
| **DeltaDock** | 2.97 | 47.4 | 66.9 | 66.7 | 83.2 | **40.8** | 61.3 | 60.6 | **78.9** |

[1] The time of consumption is denoted with * if it only consumes CPU.
[2] All results of baselines are taken from [11] for fair comparison.

### 3.4.2 Bi-EGMN

We design a physics-informed loss function for the Bi-EGMN module for training. The coordinates $X^{a,\mathcal{L}}$ and $X^{r,\mathcal{L}}$ output by the last layer of atomic level and residue level are both employed in the computation of this loss. Formally, the loss function can be expressed as follows:

$$L = L_{inter} + \lambda_1 L_{intra} + \lambda_2 L_{vdw} + \lambda_3 L_{bound}, \tag{11}$$

where $\lambda$ are weight hyper-parameters. Among the four components, inter-distance map loss $L_{inter}$ is responsible for the RMSD accuracy. Other three items, namely intra-distance map loss $L_{intra}$, vdw constraint loss $L_{vdw}$, and bound matrix constraint loss $L_{bound}$ are employed for physical validity. When training and inferencing, we follow previous work [33] and employ the recycling strategy (details see Appendix.B.3.2).

## 4 Experiments

### 4.1 Settings

**Dataset.** We conduct experimetns on PDBbind [34] v2020 and PoseBusters [16] datasets in this work. Our model is trained on the PDBbind dataset, where the training, validation, and testing set are constructed based on the time split strategy used in previous work [11]. PoseBusters, which contains 428 carefully selected data released from 1 January 2021 to 30 May 2023, is directly adopted to evaluate the ability to predict physically valid poses.

**Evaluation.** Root-mean-square-deviation (RMSD) and centroid distance (CD) are used to evaluate the docking accuracy of different docking methods, and the PoseBusters [16] test suite is employed to evaluate the performance of predicting physically valid poses. Additionally, as pocket prediction plays an important role in our framework, the distance between the center of the predicted pocket and the center of the ground-truth ligand structure (DCC), and the volume coverage rate (VCR) are employed to evaluate the pocket prediction accuracy (details in Appendix.B).

### 4.2 Overall Performance on the PDBbind

We first assess the comprehensive performance of DeltaDock on the PDBbind dataset, encompassing both blind docking and site-specific docking settings.

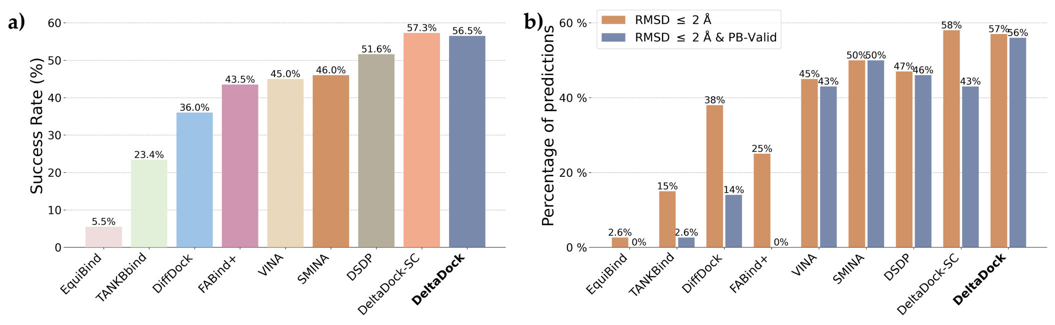

Figure 2: Site-specific docking performance. (a) Overall Performance of different methods on the PDBbind test set. The search space was delineated by extending the minimum and maximum of the x, y, and z coordinates of the ligand by 4 Å respectively. For TANKBind, we directly supply the protein block with a radius of 20 Å centered around the ground-truth ligand center to the model. (b) Overall performance of different methods on the PoseBusters dataset. (c) A waterfall plot for illustrating the PoseBusters tests as filters for both DeltaDock and DeltaDock-SC predictions. The evaluation results for DeltaDock are denoted above the lines, while those for DeltaDock-SC are annotated below.

### 4.2.1 Blind Docking

As demonstrated in Table.1, DeltaDock outperforms all baseline methods. Specifically, DeltaDock achieves a remarkable success rate of 47.4% (where RMSD < 2.0 Å), surpassing the previous SOTA GDL method, DiffDock, which has a success rate of 36.0%. Recent GPU-accelerated docking methods have also made significant progress in blind docking. However, when compared to DSDP, which is the top-performing sampling-based method in the PDBbind test set, DeltaDock still exhibits superior performance across all metrics. Notably, as elucidated in Section 3.3, DeltaDock employs the same sampling algorithm as DSDP for generating the initial structure. Yet, our framework allows DeltaDock to significantly outperform DSDP.

Beyond accuracy, efficiency is a critical performance measure for molecular docking methods. As indicated in Table 1, DeltaDock maintains a competitive level of efficiency, despite the inclusion of an energy minimization operation to enhance accuracy and reliability. Molecular docking methods invariably face a trade-off between efficiency and accuracy. However, the data presented in Table 1 suggest that DeltaDock could serve as a viable tool for practical applications, balancing these two crucial aspects effectively.

### 4.2.2 Site-specific Docking

Most existing GDL methods, such as DiffDock and EquiBind, are primarily designed for blind docking scenarios and are not inherently suited for site-specific docking tasks. However, DeltaDock seamlessly integrates blind docking and site-specific docking settings. In this context, the pocket is directly provided, eliminating the need for pocket selection via CPLA. The performance of DeltaDock in site-specific docking is illustrated in Fig.2. When supplied with predefined binding sites, traditional sampling methods exhibit a significant improvement in results. For instance, the docking success rate of VINA escalates from 10.3% to 45.0%. Despite this enhancement, DeltaDock consistently surpasses all baselines. Previous research suggested that while GDL docking methods excel at pocket searching, traditional methods tend to outperform GDL models in site-specific docking tasks [35]. However, as evidenced by the results presented in Table.1 and Fig.2, DeltaDock exhibits superior performance in both blind and site-specific docking scenarios, demonstrating its versatility and robustness in handling diverse docking settings.

## 4.3 Evaluation of Pose Validity

We further investigate DeltaDock's ability to predict physically valid structures by employing the PoseBusters test suite, as designed by Buttenschoen et al. [16]. In addition to the RMSD between predicted and ground-truth poses, the test suite incorporates 18 checks, encompassing chemical validity and consistency, intramolecular validity, and intermolecular validity. When physical validity

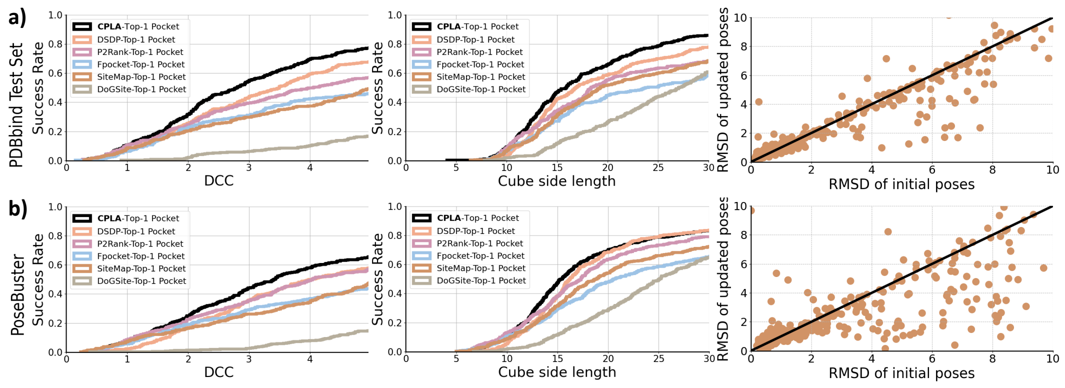

Figure 3: Further analysis on the (a) PDBbind and (b) PoseBusters dataset. Left: DCC cumulative curve of top-1 pockets. Middle: VCR cumulative curve of top-1 pockets. Right: Scatter plot of RMSD of initial and updated poses. All experiments are conducted in the blind docking setting.

is considered, the docking success rates of traditional sampling methods remain stable, while the performance of previous geometric deep learning methods significantly declines, especially for TANKBind, DeepDock, and Uni-Mol. The DeltaDock-SC variant, even without the application of the fast structure correction step, shows significant improvement over previous methods. These results substantiate DeltaDock's capacity to predict physically valid structures, thereby affirming its reliability for practical applications.

### 4.4 Further Analysis

#### 4.4.1 Pocket-ligand Alignment and Iterative Refinement

Beyond the overall docking performance, the pocket-ligand alignment and iterative refinement results are explored (Fig. 3). As depicted in the figure, CPLA predicts significantly more accurate pockets than other methods and Bi-EGMN can diminish the discrepancy between ground-truth structures and input structures. Generally, the PDBbind test set poses a more significant challenge to Bi-EGMN than the PoseBusters dataset. And for CPLA, PoseBusters dataset is more challenging otherwise. The consistent good performance on the two datasets demonstrates the effectiveness and generalization capacity of CPLA and Bi-EGMN.

#### 4.4.2 Ablation Studies

In this section, ablation studies are conducted to assess the contributions of different components. We first ablate the whole CPLA or Bi-EGMN, and then the residue-level or the atom-level in Bi-EGMN (see Appendix. B.4 for implement details). As illustrated in Table 2, it becomes clear that each component, encompassing CPLA and the bi-level strategy in Bi-EGMN, plays a significant role in enhancing the overall performance of DeltaDock. Due to the space limitation, a full ablation study can be found in Appendix. C.3.

Table 2: Results of ablation study.

| Method | RMSD % below 2 Å | |
|---|---|---|
| | PDBbind | PoseBusters |
| **DeltaDock** | **47.4** | **49.3** |
| w/o CPLA | 41.2 | 43.7 |
| w/o Bi-EGMN | 44.6 | 41.8 |
| w/o Residue Level | 44.6 | 44.4 |
| w/o Atom Level | 44.6 | 42.1 |

## 5 Conclusion

In this work, we proposed DeltaDock, a unified framework for accurate, efficient, and physically reliable molecular docking. DeltaDock was a two-stage docking framework, consisting of pocket prediction and site-specific docking. We innovatively reframed the pocket prediction task as a pocket-ligand alignment problem and then followed a hybrid strategy to jointly utilize both GDL and physics-informed traditional algorithms for site-specific docking. Comprehensive experiments demonstrated the superior performance of DeltaDock. Notably, in the blind docking setting, DeltaDock achieved a 31% relative improvement over the docking success rate compared with the previous state-of-the-art GDL model. We hope this work will further facilitate the broad application and continued development of the molecular docking framework.

## 6 Acknowledgements

We extend our gratitude to the reviewers for their valuable and insightful feedback, which significantly improved this work. We are also grateful to Lixue Cheng from Microsoft Research for her helpful suggestions and comments. This research was supported by grants from the National Natural Science Foundation of China (No.62406303, No.623B2095), the National Key R&D Program of China (No.2023YFF1205103), the Anhui Provincial Natural Science Foundation (No.2308085QF229), and the Fundamental Research Funds for the Central Universities.

## References

[1] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. In *ArXiv*, 2021.

[2] Zaixi Zhang, Zepu Lu, Zhongkai Hao, Marinka Zitnik, and Qi Liu. Full-atom protein pocket design via iterative refinement. In *NeurIPS'23*, 2023.

[3] Zaixi Zhang and Qi Liu. Learning subpocket prototypes for generalizable structure-based drug design. In *ICML'24*, 2023.

[4] Xing Du, Yi Li, Yuan-Ling Xia, Shi-Meng Ai, Jing Liang, Peng Sang, Xing lai Ji, and Shu-Qun Liu. Insights into protein–ligand interactions: Mechanisms, models, and methods. *International Journal of Molecular Sciences*, 17:144, 2016.

[5] Shuangli Li, Jingbo Zhou, Tong Xu, Liang Huang, Fan Wang, Haoyi Xiong, Weili Huang, Dejing Dou, and Hui Xiong. Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity. In *KDD'21*, 2021.

[6] Jiankun Lyu, Sheng Wang, Trent E. Balius, Isha Singh, Anat Levit, Yurii S. Moroz, Matthew J. O'Meara, Tao Che, Enkhjargal Algaa, Kateryna A Tolmachova, Andrey A. Tolmachev, Brian K. Shoichet, Bryan L. Roth, and John J. Irwin. Ultra-large library docking for discovering new chemotypes. *Nature*, 566:224 – 229, 2019.

[7] Brian Joseph Bender, Stefan Gahbauer, Andreas Luttens, Jiankun Lyu, Chase M Webb, Reed M. Stein, Elissa A. Fink, Trent E. Balius, Jens Carlsson, John J. Irwin, and Brian K. Shoichet. A practical guide to large-scale docking. *Nature protocols*, page 4799–4832, 2021.

[8] Hannes Stärk, Octavian-Eugen Ganea, Lagnajit Pattanaik, Regina Barzilay, and T. Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. In *ICML'22*, 2022.

[9] Gabriele Corso, Hannes Stärk, Bowen Jing, Regina Barzilay, and Tommi S. Jaakkola. Diffdock: Diffusion steps, twists, and turns for molecular docking. In *ICLR'23*, 2023.

[10] Zaixi Zhang, Jiaxian Yan, Qi Liu, and Enhong Chen. A systematic survey in geometric deep learning for structure-based drug design. *ArXiv*, abs/2306.11768, 2023.

[11] Qizhi Pei, Kaiyuan Gao, Lijun Wu, Jinhua Zhu, Yingce Xia, Shufang Xie, Tao Qin, Kun He, Tie-Yan Liu, and Rui Yan. Fabind: Fast and accurate protein-ligand binding. In *NeurIPS'23*, 2023.

[12] Yangtian Zhang, Huiyu Cai, Chence Shi, and Jian Tang. E3bind: An end-to-end equivariant network for protein-ligand docking. In *ICLR'23*, 2023.

[13] Nafisa Hassan, Amr Alhossary, Yuguang Mu, and C. Kwoh. Protein-ligand blind docking using quickvina-w with inter-process spatio-temporal integration. *Scientific Reports*, 7, 2017.

[14] 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 *ICLR'23*, 2023.

[15] Jintao Zhu, Zhonghui Gu, Jianfeng Pei, and Luhua Lai. Diffbindfr: An se(3) equivariant network for flexible protein-ligand docking. In *ArXiv*, 2023.

[16] Martin Buttenschoen, Garrett M. Morris, and Charlotte M. Deane. Posebusters: Ai-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 2023.

[17] Danny Reidenbach and Aditi S. Krishnapriyan. Coarsenconf: Equivariant coarsening with aggregated attention for molecular conformer generation. In *ArXiv*, 2023.

[18] Oleg Trott and Arthur J. Olson. Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31:455–461, 2010.

[19] David Ryan Koes, Matthew P. Baumgartner, and Carlos J. Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of chemical information and modeling*, 53 8:1893–904, 2013.

[20] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer New York, NY, 2000.

[21] Ji Ding, Shi xiong Tang, Zheming Mei, Lingyue Wang, Qinqin Huang, Haifeng Hu, Ming Ling, and Jiansheng Wu. Vina-gpu 2.0: Further accelerating autodock vina and its derivatives with graphics processing units. *Journal of chemical information and modeling*, 63:1982–1998, 2023.

[22] Yuejiang Yu, Chun Cai, Jiayue Wang, Zonghua Bo, Zhengdan Zhu, and Hang Zheng. Uni-dock: Gpu-accelerated docking enables ultralarge virtual screening. *Journal of chemical theory and computation*, 19:3336–3345, 2023.

[23] Yupeng Huang, Hong Zhang, Siyuan Jiang, Dajiong Yue, Xiaohan Lin, Jun Zhang, and Yi Qin Gao. Dsdp: A blind docking strategy accelerated by gpus. *Journal of chemical information and modeling*, 63:4355–4363, 2023.

[24] Vincent Le Guilloux, Peter Schmidtke, and Pierre Tufféry. Fpocket: An open source platform for ligand pocket detection. *BMC Bioinformatics*, 10:168 – 168, 2009.

[25] Radoslav Krivák and David Hoksza. P2rank: machine learning based tool for rapid and accurate prediction of ligand binding sites from protein structure. *Journal of Cheminformatics*, 10:39, 2018.

[26] Jeevan Kandel, Hilal Tayara, and Kil to Chong. Puresnet: prediction of protein-ligand binding sites using deep residual neural network. *Journal of Cheminformatics*, 13:65, 2021.

[27] Greg Landrum, Paolo Tosco, Brian Kelley, Ric, sriniker, gedeck, Riccardo Vianello, NadineSchneider, Eisuke Kawashima, Andrew Dalke, David Cosgrove, Dan N, Gareth Jones, Brian Cole, Matt Swain, Samo Turk, AlexanderSavelyev, Alain Vaucher, Maciej Wójcikowski, Ichiru Take, Daniel Probst, Kazuya Ujihara, Vincent F. Scalfani, guillaume godin, Axel Pahl, Francois Berenger, JLVarjo, strets123, JP, and DoliathGavid. rdkit/rdkit: 2022_03_4 (q1 2022) release, July 2022.

[28] Zhaoping Xiong, Dingyan Wang, Xiaohong Liu, Feisheng Zhong, Xiaozhe Wan, Xutong Li, Zhaojun Li, Xiaomin Luo, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Pushing the boundaries of molecular representation for drug discovery with graph attention mechanism. *Journal of medicinal chemistry*, 63:8749–8760, 2020.

[29] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael J. L. Townshend, and Ron O. Dror. Learning from protein structure with geometric vector perceptrons. In *ICLR '21*, 2020.

[30] Wei Lu, Qifeng Wu, Jixian Zhang, Jiahua Rao, Chengtao Li, and Shuangjia Zheng. Tankbind: Trigonometry-aware neural networks for drug-protein binding structure prediction. In *NeurIPS'22*, 2022.

[31] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML'17*, 2017.

[32] Peter K. Eastman, Jason M. Swails, John D. Chodera, Robert T. McGibbon, Yutong Zhao, Kyle A. Beauchamp, Lee-Ping Wang, Andrew C. Simmonett, Matthew P. Harrigan, Chaya D. Stern, Rafal P. Wiewiora, Bernard R. Brooks, and Vijay S. Pande. Openmm 7: Rapid development of high performance algorithms for molecular dynamics. *PLoS Computational Biology*, 13, 2016.

[33] John M. Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David A. Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589, 2021.

[34] Zhihai Liu, Minyi Su, Li Han, Jie Liu, Qifan Yang, Yan Li, and Renxiao Wang. Forging the basis for developing protein-ligand interaction scoring functions. *Accounts of chemical research*, 50 2:302–309, 2017.

[35] Yuejiang Yu, Shuqi Lu, Zhifeng Gao, Hang Zheng, and Guolin Ke. Do deep learning models really outperform traditional approaches in molecular docking? In *ArXiv*, 2023.

[36] Limei Wang, Haoran Liu, Yi Liu, Jerry Kurtin, and Shuiwang Ji. Learning hierarchical protein representations via complete 3d graph networks. In *ICLR'23*, 2022.

[37] Tianfan Fu and Jimeng Sun. Sipf: Sampling method for inverse protein folding. In *KDD'22*, 2022.

[38] David Dohan, Andreea Gane, Maxwell L. Bileschi, David Belanger, and Lucy J. Colwell. Improving protein function annotation via unsupervised pre-training: Robustness, efficiency, and insights. In *KDD'21*, 2021.

[39] Joel Graef, Christiane Ehrt, and Matthias Rarey. Binding site detection remastered: Enabling fast, robust, and reliable binding site detection and descriptor calculation with dogsite3. *Journal of Chemical Information and Modeling*, 63(10):3128–3137, 2023.

[40] Tom Halgren. New method for fast and accurate binding-site identification and analysis. *Chemical biology & drug design*, 69(2):146–148, 2007.

[41] Thomas A Halgren. Identifying and characterizing binding sites and assessing druggability. *Journal of chemical information and modeling*, 49(2):377–389, 2009.

[42] Noel M. O'Boyle, Michaela S. Banck, Craig A. James, Chris Morley, Tim Vandermeersch, and Geoffrey R. Hutchison. Open babel: An open chemical toolbox. *Journal of Cheminformatics*, 3:33 – 33, 2011.

[43] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[44] Oscar Méndez-Lucio, Mazen Ahmad, Ehecatl Antonio del Rio-Chanona, and Jörg Kurt Wegner. A geometric deep learning approach to predict binding conformations of bioactive molecules. *Nat. Mach. Intell.*, 3:1033–1039, 2021.

[45] Kaiyuan Gao, Qizhi Pei, Jinhua Zhu, Tao Qin, Kun He, Tie-Yan Liu, and Lijun Wu. Fabind+: Enhancing molecular docking through improved pocket prediction and pose generation. *ArXiv*, abs/2403.20261, 2024.

[46] Rocco Meli and Philip Charles Biggin. spyrmsd: symmetry-corrected rmsd calculations in python. *Journal of Cheminformatics*, 12, 2020.

[47] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR'15*, 2015.

[48] Bowen Gao, Bo Qiang, Haichuan Tan, Minsi Ren, Yinjun Jia, Minsi Lu, Jingjing Liu, Weiying Ma, and Yanyan Lan. Drugclip: Contrastive protein-molecule representation learning for virtual screening. In *NeurIPS '23*, 2023.

[49] Greg Landrum et al. Rdkit: A software suite for cheminformatics, computational chemistry, and predictive modeling. *Greg Landrum*, 2013.

[50] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J Ballard, Joshua Bambrick, Sebastian W Bodenstein, David A Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander Imani Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B Fuchs, Hannah Gladman, Rishub Jain, Yousuf A Khan, Caroline M R Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Vic-613 tor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold3. *Nature*, 2024.

## A More Detailed Descriptions

### A.1 Dataset Preprocessing

We follow the time split strategy used in previous work [8, 30, 11] to split the dataset to construct the train, validation, and test set. All compounds discovered in or after 2019 are in the test and validation sets, and only those found before 2019 are in the training set. The training set, validation set, and test set have 17,299, 968, and 363 complexes, respectively. The overall performance of docking methods is evaluated on the time spit test set following previous works. In this work, we only select the protein chains within 10 Å to the ligand structure.

### A.2 Dataset Statistics

Proteins are inherently macromolecules composed of multiple chains, with each chain potentially containing hundreds or even thousands of residues [36, 37, 38]. In Table.3, we statistically analyze the PDBbind time-split test set and count atom numbers in proteins. Notably, it can be observed that the number of atoms escalates substantially as the cutoff value increases.

Table 3: Statistics of the PDBbind time split test set.

| Data | Average | | Maximum | |
|---|---|---|---|---|
| | Number of $C_\alpha$ | Number of atoms | Number of $C_\alpha$ | Number of atoms |
| Entire protein structure | 322 | 2,536 | 1,488 | 11,697 |
| Structure within 40.0 Å cubic box centered on the ligand | 179 | 1,602 | 400 | 3,055 |
| Structure within 15.0 Å from ligand | 111 | 1,050 | 213 | 1,944 |
| Structure within 12.0 Å from ligand | 73 | 740 | 164 | 1,582 |
| Structure within 8.0 Å from ligand | 30 | 379 | 75 | 986 |
| Structure within 6.0 Å from ligand | 16 | 207 | 45 | 548 |

### A.3 Example of Large Pocket

Large pockets that consist of several sub-pockets generally exist. For example, the main protease of SARS-CoV-2 (Fig. 4).

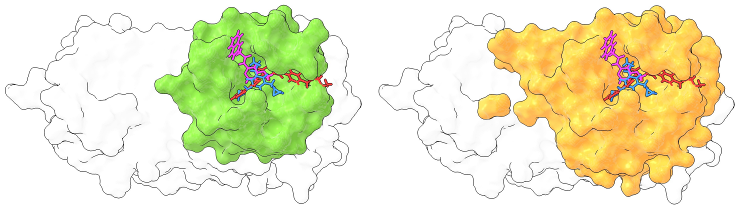

Figure 4: The main protease of SARS-CoV-2 is depicted by the white surface. The ligand structures in pink, blue, and red correspond to PDB 5RGY, 7AQJ, and 7JU7, respectively. **Left:** The green pocket, a protein structure truncated to within 12.0 Å of the blue structure, is insufficient to encompass the pocket structure necessary for predicting the red structure. **Right:** The orange pocket, truncated within a 40.0 Å box utilized by DeltaDock, is ample to cover the entire pocket.

### A.4 Analysis of Existing Pocket Prediction Methods

As depicted in Fig.5, existing pocket prediction methods generally achieve a hit rate of approximately 70%-80%, where the distance between the predicted pocket center and ligand center (DCC) is less than 5.0 Å. Notably, when leveraging combined predictions from multiple methods, the hit rate significantly increases to nearly 95%. Motivated by this observation, DeltaDock begins with a ready-to-dock ligand and a candidate pocket set derived from a suite of existing pocket prediction models.

We further statistics how many pockets these methods predict in Fig. 6. We observe that Fpocket [24], and DoGSite3 [39] output much more pockets than DSDP [23], P2Rank [25], and SiteMap [40, 41].

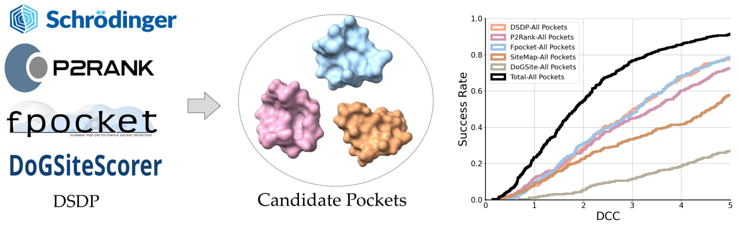

Figure 5: Performance of different pocket prediction methods on the PDBbind test set. The hit rate is significantly improved by ensembling the predicted pockets from various methods.

Combining information from Fig. 6 and Fig.5, it is evident that the pockets predicted by DSDP and P2rank are highly druggable. Other methods, in contrast, tend to predict many non-druggable pockets.

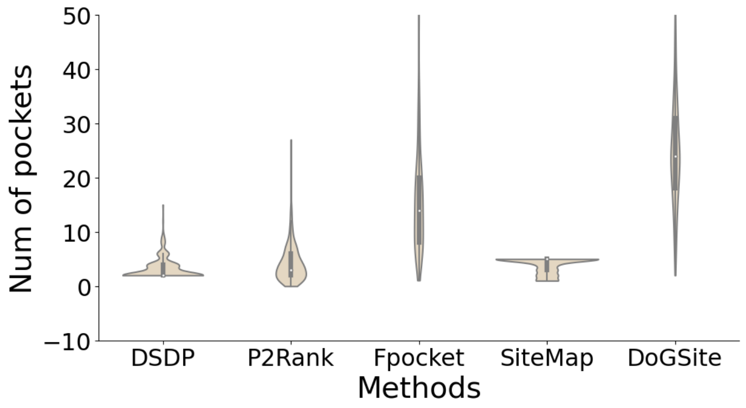

Figure 6: Pocket numbers violin plot of different methods. Pocket prediction methods generally predict a series of druggable pockets.

## A.5 Efficiency Comparison between SMINA and OpenMM

For AI-based structure prediction methods, including AlphaFold2 [33], it is common practice to employ energy minimization methods for post-processing to ensure the physical validity of the predicted structures. While specialized methods like OpenMM are available for energy minimization, we opted not to use them due to computational efficiency considerations. Specifically, we found that SMINA, which is typically known as a docking method, requires only approximately 0.4 seconds for energy minimization. This is significantly faster than methods like OpenMM, which can take several minutes to tens of minutes per protein-ligand pair, as illustrated in the Table. 4 below.

For molecular docking, efficiency is crucial, and specialized methods such as OpenMM can be excessively time-consuming. What's more, it is important to note that SMINA, although generally regarded as a docking method, is not employed for docking in our workflow but rather utilized in its minimization mode for energy minimization.

Table 4: Efficiency Comparison between SMINA and OpenMM.

| **Methods** | **Time (per protein-ligand pair)** |
|---|---|
| SMINA | about 0.4 seconds |
| OpenMM | several minutes to tens of minutes |

### A.6 Graph Construction

**Ligand Graph.** The input ligand $L$ is first represented as a ligand graph $\mathcal{G}^{\mathcal{L}} = (\mathcal{V}^{\mathcal{L}}, \mathcal{E}^{\mathcal{L}})$, where $\mathcal{V}^{\mathcal{L}}$ is the node set and node $i$ represents the $i$-th atom in the ligand. In this work, RdKit [27] is employed to generate a 3D initial conformer of the input ligand. Each node $v_i^{\mathcal{L}}$ is also associated with an atom coordinate $x_i^{\mathcal{L}}$ retrieved from the individual ligand structure $\mathcal{P}$ and an atom feature vector $h_i^{\mathcal{L}}$. The edge set $\mathcal{E}^{\mathcal{L}}$ is constructed according to the spatial distances among atoms. More formally, the edge set is defined to be:

$$\mathcal{E}^{\mathcal{L}} = \left\{(i,j) : |x_i^{\mathcal{L}} - x_j^{\mathcal{L}}|^2 < cut^{\mathcal{L}}, \forall i,j \in \mathcal{V}^{\mathcal{L}}\right\}, \tag{12}$$

where $cut^{\mathcal{L}}$ is a distance threshold, and each edge $(i,j) \in \mathcal{E}^{\mathcal{L}}$ is associated with an edge feature vector $e_{ij}^{\mathcal{L}}$. The node and edge features are obtained by RDKit [27] in the CPLA. And in the Bi-EGMN, they are achieved by OpenBabel [42]

**Protein Atomic Graph.** The protein atomic graph $\mathcal{G}^{\mathcal{P}}$ is constructed in the same way as the ligand graph.

**Protein Residue Graph.** For protein residue Graph $\mathcal{G}^{\mathcal{P}*} = (\mathcal{V}^{\mathcal{P}*}, \mathcal{E}^{\mathcal{P}*})$, $\mathcal{V}^{\mathcal{P}*}$ is the node set and the node $i$ represents the $i$-th residue in the protein. Each node $v_i^{\mathcal{P}*}$ is also associated with an $C_\alpha$ coordinate of the $i$-th residue $x_i^{\mathcal{P}*}$ retrieved from the individual protein structure and a residue feature vector $h_i^{\mathcal{P}*}$. The edge set $\mathcal{E}^{\mathcal{P}*}$ is constructed according to the spatial distances among atoms. More formally, the edge set is defined to be:

$$\mathcal{E}^{\mathcal{P}*} = \left\{(i,j) : |x_i^{\mathcal{P}*} - x_j^{\mathcal{P}*}|^2 < cut^{\mathcal{P}*}, \forall i,j \in \mathcal{V}^{\mathcal{P}*}\right\}, \tag{13}$$

where $cut^{\mathcal{P}*}$ is a distance threshold, and each edge $(i,j) \in \mathcal{E}^{\mathcal{P}*}$ is associated with an edge feature vector $e_{ij}^{\mathcal{P}*}$. The edge features are obtained following [9]. As for the node features, they are extracted from the protein language model ESM2-3B [43] in CPLA. While in Bi-EGMN, they are obtained following [8].

## B More Detailed Experimental Settings

### B.1 Baselines

For molecular docking, GDL methods, EquiBind [8], TANKBind [30], DiffDock [9], DeepDock [44], Uni-Mol [14], FABind [11], and FABind+ [45] and traditional sampling methods, VINA [18], SMINA [19], and DSDP [23] are used as baselines. According to their performance on both PDBbind and Posebusters, DiffDock is the previous SOTA GDL docking method, and we thus mainly focus on the comparison with DiffDock. For pocket prediction, DSDP, P2Rank [25], Fpocket [24], SiteMap [40, 41], and DoGSite3 [39] are compared.

### B.2 Evaluation Metric

For blind docking and site-specific docking, RMSD and centroid distance are used to evaluate different methods, the formal definitions of these two metrics are:

$$RMSD = \sqrt{\frac{1}{|V|}\sum_{i=1}^{|V|}(x_i^{\mathcal{L}} - \hat{x}_i^{\mathcal{L}'})^2}, \tag{14}$$

$$Centroid = |\frac{1}{|V|}\sum_{i=1}^{|V|} x_i^{\mathcal{L}} - \frac{1}{|V|}\sum_{i=1}^{|V|}\hat{x}_i^{\mathcal{L}'}|, \tag{15}$$

where $x_i^{\mathcal{L}}$ is the ground truth coordinate of $i$-th ligand atom and $\hat{x}_i^{\mathcal{L}'}$ is the predicted coordinates. In alignment with previous studies [15, 9], for blind docking, the RMSD is directly computed. However, in the case of site-specific docking, the RMSD is calculated utilizing the spyrmsd [46].

For pocket prediction, the DCC metric is defined as:

$$DCC = |\hat{\varsigma} - \frac{1}{|V|}\sum_{i=1}^{|V|}\hat{x}_i^{\mathcal{L}'}|, \tag{16}$$

where $\hat{\varsigma}$ is the predicted pocket center. As for the VCR metric [23], we calculate the cube side length of a cube box centered on the pocket that can cover the whole ligand structure.

### B.3 Training and inference

Our models are trained using NVIDIA A100-PCIE-40GB GPUs. Training the CPLA on a single GPU takes approximately 2 hours, while the Bi-EGMN requires about 48 hours on 4 GPUs. To determine the hyperparameters, we performed a grid search, as outlined in Table 5 and Table 6.

#### B.3.1 CPLA

**Basic Settings.** The model was trained employing the Adam optimizer [47] with an initial learning rate of $0.0003$ and an $L_2$ regularization factor of $10^{-6}$. The learning rate was scaled down by 0.6 if no drop in training loss was observed for 10 consecutive epochs. The number of training epochs was set to 20 with an early stopping rule of 10 epochs if no improvement in the validation performance was observed.

**Candidate Pockets Generation.** For CPLA, we consider two methods to generate candidate pockets: DSDP, and P2Rank. These methods were selected over others, such as SiteMap. Initially, we intended to incorporate all available methods to construct the candidate pockets. However, the results were unsatisfactory. This could be attributed to the issue of hard negative samples. CPLA employs contrastive learning, where the quality of hard negative sample selection directly impacts the training performance. In this context, hard negative samples represent highly druggable pockets that are not the target pocket. As illustrated in Fig. 6 and Fig.5, the pockets predicted by DSDP and P2rank are highly druggable. In contrast, other methods tend to predict non-druggable pockets. The result in Table. 7 demonstrates that introducing FPocket impairs the training quality. Consequently, we opted to solely use DSDP and P2rank.

**Pocket Augmentation.** Given a candidate pockets set $S = \{\varsigma_1, \varsigma_2, ...\}$, we establish a maximum pocket number, $N_{max}$, to construct negative pockets for data augmentation. If $|S| >= N_{max}$, we select the top-$N_{max}$ pockets in the sort of DSDP, P2Rank accordingly. If $|S| < N_{max}$, we randomly select $(N_{max} - |S|)$ $C_\alpha$ atoms that are more than 20.0 Å from the ligand geometric center to construct negative pocket centers. This data augmentation is only applied in the training phase.

**Ligand Conformation Augmentation.** During the CPLA training, we further considered the issue of the native binding mode. As the native binding mode (i.e., the co-crystal structure) of a given molecule is unknown in practical scenarios, we aim to train a pose-robust CPLA model. To achieve this, we adjusted the rotatable bond angles of the co-crystal molecule structure in each epoch during training. Therefore, the molecule poses in each epoch are perturbed and different.

**Other Training Object.** We have considered using cross-protein loss for training, where the ground truth pockets and ligands from the same protein-ligand pairs are considered positive samples, and those from different protein-ligand pairs are treated as negative samples. Although this loss has been utilized in previous work for virtual screening [48], it was found to be unsuitable for our model.

#### B.3.2 Bi-EGMN

**Basic Settings.** The Adam optimizer [47], characterized by an initial learning rate of $10^{-3}$ and an $L_2$ regularization factor of $10^{-6}$, is employed for training Bi-EGMN. The learning rate was scaled down by 0.6 if no drop in training loss was observed for 10 consecutive epochs. The number of training epochs was set to 1000 with an early stopping rule of 40 epochs if no improvement in the validation performance was observed.

**Training Object.** The loss function can be written as:

$$L = L_{inter} + \lambda_1 L_{intra} + \lambda_2 L_{vdw} + \lambda_3 L_{bound}. \tag{17}$$

As introduced before, the inter-distance map loss $L_{inter}$ is responsible for the RMSD accuracy. Other three items, namely intra-distance map loss $L_{intra}$, vdW constraint loss $L_{vdw}$, and bound matrix constraint loss $L_{bound}$ are employed for physical validity.

Table 5: The hyperparameter options we searched through for CPLA. The final parameters are marked in **bold**.

| Parameter | Search Sapce |
|---|---|
| Number of layers | 2, **3**, 4 |
| Batch Size | 8, **16**, 32, 64, 128 |
| Dropout | **0.1** |
| Learning rate | 0.003, 0.001, **0.0003**, 0.0001 |
| Max pocket number for training | Null, 16, **32**, 64, 128 |
| Pocket used for training | **[DSDP, P2Rank]** |
| Training loss | **Intra-protein**, Cross-protein |
| ESM2-3B embedding | **True**, False |
| AFP hidden dimension | 64, **128**, 256 |
| GVP node scalar hidden dimension | 32, **64**, 128 |
| GVP node vector hidden dimension | 12, **16**, 32 |
| GVP edge scalar hidden dimension | **32**, 64, 128 |
| GVP edge vector hidden dimension | **12**, 16, 32 |

The two distance map losses can be formally expressed as:

$$L_{inter} = \sum_{i \in \mathcal{V}_{\mathcal{L}}} \sum_{j \in \mathcal{V}_{\mathcal{P}}} ||d_{ij}^{pred} - d_{ij}^{gt}||, \; L_{intra} = \sum_{i \in \mathcal{V}_{\mathcal{L}}} \sum_{j \in \mathcal{V}_{\mathcal{L}}} ||d_{ij}^{pred} - d_{ij}^{gt}||, \tag{18}$$

where predicted distance $d_{ij}^{pred} = ||x_i^{pred} - x_j^{pred}||$ and ground-truth distance $d_{ij}^{gt} = ||x_i^{gt} - x_j^{gt}||$ between node $i$ and $j$ are calculated based on node coordinates.

The other two physics-informed losses can be formally expressed as:

$$L_{vdw} = \sum_{i \in \mathcal{V}_{\mathcal{L}}} \sum_{j \in \mathcal{V}_{\mathcal{P}}} max(d_{ij}^{vdw} - d_{ij}^{pred}, 0), \tag{19}$$

$$L_{bound} = \sum_{i \in \mathcal{V}_{\mathcal{L}}} \sum_{j \in \mathcal{V}_{\mathcal{L}}} max(d_{ij}^{bd,low} - d_{ij}^{pred}, 0) + max(d_{ij}^{pred} - d_{ij}^{bd,up}, 0), \tag{20}$$

where the vdW distance $d_{ij}^{vdw} = 0.75(r_i^{vdw} + r_j^{vdw})$ is calculated based on node van der Waals radii $r^{vdw}$. As for the lower bound distance $d_{ij}^{bd,low}$ and upper bound distance $d_{ij}^{bd,up}$, they are determined based on the bound matrix generated by RDKit [49] following [16].

**Initial Poses Augmentation.** In the training phase of the Bi-EGMN, initial pose augmentation is employed. The initial poses utilized for training are sampled based on the ground truth pocket. An adaptive box is defined through a two-step process: (1) the minimum and maximum of the x, y, and z coordinates of the ligand are extended by 4 Å each; (2) if the box size is less than 22.5 Å after the first step, it is further extended to 22.5 Å. During the inference phase, however, the box size is fixed at 30.0 Å, deviating from the adaptive strategy employed during training. For each epoch during training, a pose is randomly selected. This pose augmentation strategy significantly amplifies the diversity of the input. As depicted in Fig.7, the sampled poses can nearly encompass the entire pocket cavity.

**Recycling.** During both training and inferencing, the recycling strategy is adopted. For training, we randomly recycle the iterative refinement process 1-3 times, and only the last cycle is used to compute the gradient. For inferencing, the recycle number is fixed to 4.

### B.4 Ablation Studies Settings

**w/o CPLA**: pockets predicted by DSDP are employed to perform the following predictions.

**w/o Bi-EGMN**: the sampled structures are directly employed as final structures to calculate metrics.

**w/o Residue Level**: the residue level is removed from Bi-EGMN.

**w/o Atom Level**: the atom level is removed from Bi-EGMN.

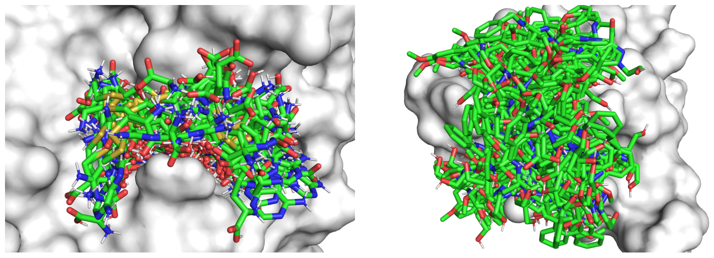

Figure 7: Initial pose augmentation. During the initial pose augmentation phase of training the Bi-EGMN, we randomly select one pose from all sampled poses for each epoch. This selection strategy ensures that the training initial poses can comprehensively cover the entire pocket.

Table 6: The hyperparameter options we searched through for Bi-EGMN. The final parameters are marked in **bold**.

| Parameter | Search Sapce |
|---|---|
| Recycle | **True**, False |
| Hidden dimension | 32, **64**, 96, 128 |
| Number of layers for each level | 4, 6, **8**, 10 |
| Batch Size | 8, 16, **32**, 64 |
| Dropout | **0.1** |
| Learning rate | **0.001** |
| Initial pose augmentation | **True**, False |
| Pose sampling box size | Adaptive, **30.0 Å** |
| CPLA pockets used for sampling | Top-1, **Top-2**, Top-3, All |
| Protein structure level | Atom level, Residue level, **Bi-level** |
| ESM2-3B embedding for residue level | True, **False** |

## C More Experimental Results

### C.1 Binding Pocket Prediction

#### C.1.1 Overall Performance on PDBbind

In addition to the overall performance presented in Fig.3, we offer a more detailed analysis in Fig.8. As can be discerned from the figure, the top-1 pockets predicted by CPLA significantly outperform those predicted by other baseline methods. Furthermore, when considering the top-2 pockets, the accuracy of pocket prediction is on par with the cumulative performance of all pockets predicted by other methods.

#### C.1.2 Results of Different Candidate Pockets

In the current framework, only DSDP and P2Rank are selected to generate candidate pockets. The motivation and analysis for this operation have been discussed before. To support this selection, we further present the experimental results of employing different candidate pockets to train CPLA in Table. 7. These results indicate that only selecting DSDP and P2Rank yields to best performance.

Table 7: Performance of employing different candidate pockets to train CPLA

| **Pockets** | **% of DCC < 4 Å** |
|---|---|
| DSDP | 64.46 |
| P2Rank | 55.37 |
| DSDP + P2Rank | 69.97 |
| DSDP + P2Rank + Fpocket | 65.84 |

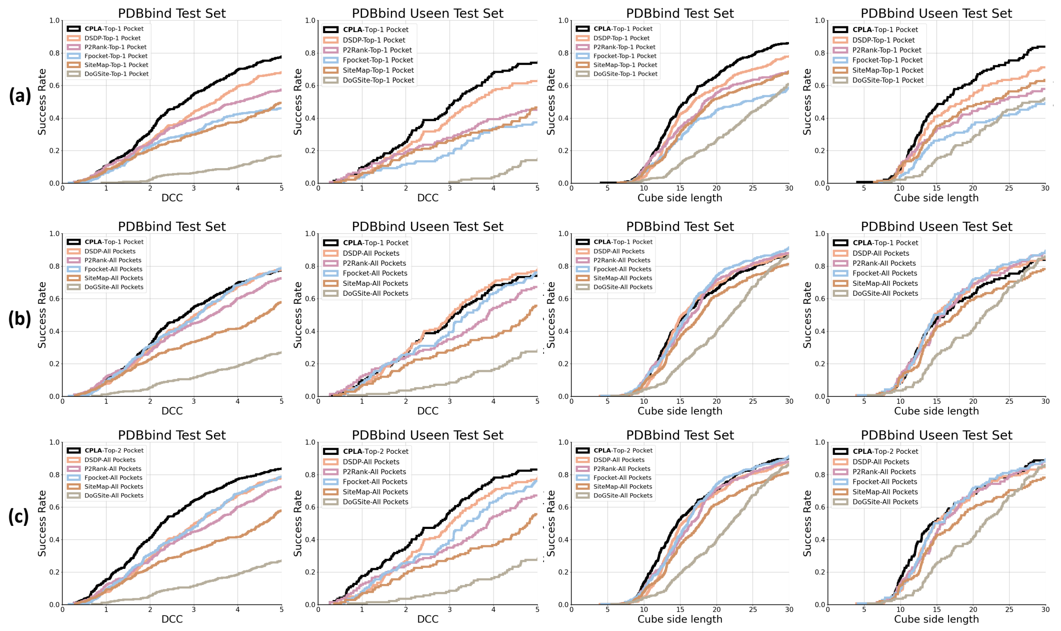

Figure 8: Performance of binding pocket prediction models on PDBbind dataset. (a) Comparison between top-1 pockets predicted by CPLA and top-1 pockets predicted by other methods. (b) Comparison between top-1 pockets predicted by CPLA and best pockets among all pockets predicted by other methods. (c) Comparison between top-2 pockets predicted by CPLA and best pockets among all pockets predicted by other methods.

### C.1.3 Influence of Ligand Conformations

When training CPLA, we employ a conformation augmentation strategy to train a pose-robust CPLA model. The provided Table. 8 illustrates CPLA's performance when presented with both a co-crystal ligand structure and an RDKit-generated ligand structure, showcasing the model's resilience to ligand poses and the effectiveness of our strategy.

Table 8: Influence of ligand conformations on CPLA

| **Input ligand pose** | **% of DCC < 4 Å** |
|---|---|
| Co-crystal | 70.25 |
| RDKit-generated | 69.97 |

### C.1.4 Comparison with FABind

Previous pocket prediction methods, such as DSDP and P2RANK, are ligand-independent. Their goal is to predict all possible binding sites. However, in molecular docking, the goal is to predict targeted binding sites. There are now methods that, like CPLA, are ligand-dependent, such as FABind. To further demonstrate the effectiveness of CPLA, a comparison on pocket prediction accuracy is conducted between FABind and CPLA as shown in Table. 9. Obviously, our model achieves a significant advantage.

Table 9: Comparison with FABind

| **Methods** | **% of DCC < 3.0 Å** | **% of DCC < 4.0 Å** |
|---|---|---|
| FABind | 42.7 | 56.5 |
| CPLA Top-1 | 54.8 | 70.0 |

## C.2 Blind Docking Performance on PoseBusters

Due to the space limitation, only site-specific docking performance on PoseBusters has been presented before. In Fig. 9, we provide the blind docking performance on PoseBusters. We observed that DeltaDock achieves a docking success rate of 48.8% even when considering the physical validity.

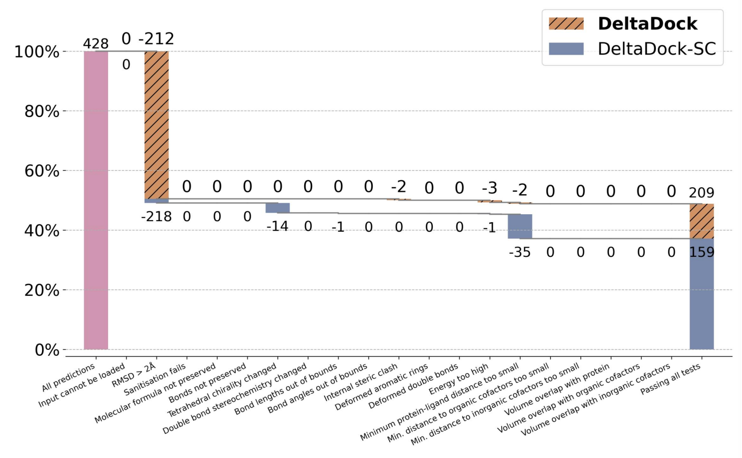

Figure 9: Blind Docking Performance on PoseBusters.

## C.3 Detailed Ablation Studies

Comprehensive ablation experiments were performed within two distinct contexts: blind docking utilizing the PDBbind dataset to assess the impact on RMSD metrics, and site-specific docking employing the PoseBusters dataset to evaluate the influence on the physical plausibility of the predicted binding poses.

### C.3.1 Ablation Studies On PDBbind

Table.10 presents more detailed ablation studies on PDBbind, including the removal of recycling, training loss components, structure correction, and structure sampling initialization. From the table, we observe that: (1) each component contributes to the good RMSD performance of our DeltaDock. (2) The training loss items and structure correction step employed for physical validity tend to decrease the RMSD performance. (3) The structure sampling algorithm used for initialization is especially important for good RMSD performance. **(4) When we train DeltaDock like previous docking methods, removing the loss items and structure correction step for physical plausibility, DeltaDock still achieves a competitive performance and outperforms all other GDL methods significantly on the test unseen set even without the using of structure sampling algorithm.** These results demonstrate the effectiveness of DeltaDock.

Table 10: Blind docking performance on the PDBbind dataset.

| Method | Time Split (363) | | | | Timesplit Unseen (142) | | | |
|---|---|---|---|---|---|---|---|---|
| | RMSD % below | | Centroid % below | | RMSD % below | | Centroid % below | |
| | 2.0Å | 5.0Å | 2.0Å | 5.0Å | 2.0Å | 5.0Å | 2.0Å | 5.0Å |
| DeltaDock | 47.4 | 66.9 | 66.7 | 83.2 | 40.8 | 61.3 | 60.4 | 78.9 |
| w/o recycle | 46.0 | 64.2 | 67.2 | 80.2 | 40.8 | 59.9 | 62.0 | 78.2 |
| w/o $L_{vdw}$ | 46.8 | 65.3 | 66.4 | 81.3 | 40.8 | 62.7 | 64.8 | 78.2 |
| w/o $L_{intra}$ | 43.5 | 64.7 | 65.0 | 84.8 | 40.1 | 58.5 | 61.3 | 81.7 |
| w/o $L_{bound}$ | 42.4 | 66.4 | 66.9 | 82.1 | 35.9 | 61.3 | 63.4 | 79.6 |
| w/o torsion alignment | 47.9 | 68.0 | 69.1 | 82.9 | 41.5 | 62.0 | 62.7 | 78.9 |
| w/o energy minimization | 46.8 | 67.8 | 70.0 | 83.2 | 40.1 | 60.6 | 65.5 | 78.8 |
| w/o structure sampling[a] | 16.0 | 53.4 | 53.2 | 80.4 | 19.0 | 51.4 | 52.1 | 73.9 |
| w/o structure sampling, and structure correction | 19.8 | 55.6 | 56.2 | 82.4 | 21.1 | 52.8 | 52.1 | 77.5 |
| w/o $L_{bound}$, $L_{intra}$, $L_{vdw}$, structure correction, structure sampling | 30.0 | 63.8 | 65.3 | 82.9 | 28.2 | 53.5 | 57.7 | 78.2 |

[a] No structure sampling means we directly put the RDKit-generated ligand structure at the center of the protein as the initial structure.

### C.3.2 Ablation Studies On PoseBusters

Fig. 10 and Fig. 11 present ablation studies on PoseBusters to explore the effect of physics-informed training items and structure correction step. From the figures, we can see that: (1) the physics-informed training items and structure correction step contribute to the good physical validity of DeltaDock. (2) Among the physics-informed training items, $L_{intra}$ is especially important for the GDL model to predict valid structures without post-processing. These results demonstrate the effectiveness of DeltaDock.

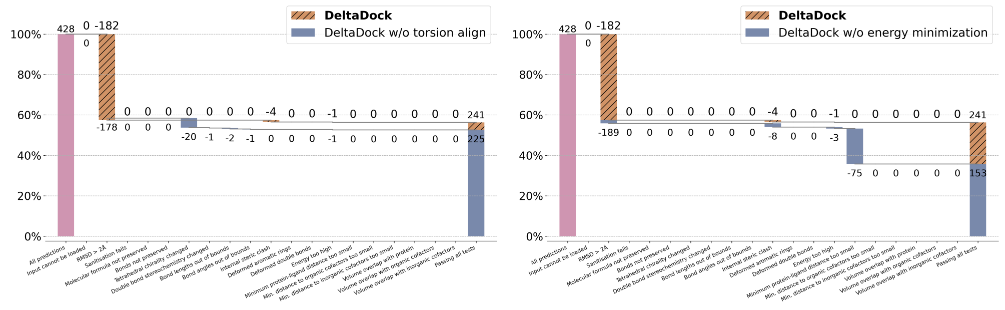

Figure 10: Site-specific docking performance on the PoseBusters dataset.

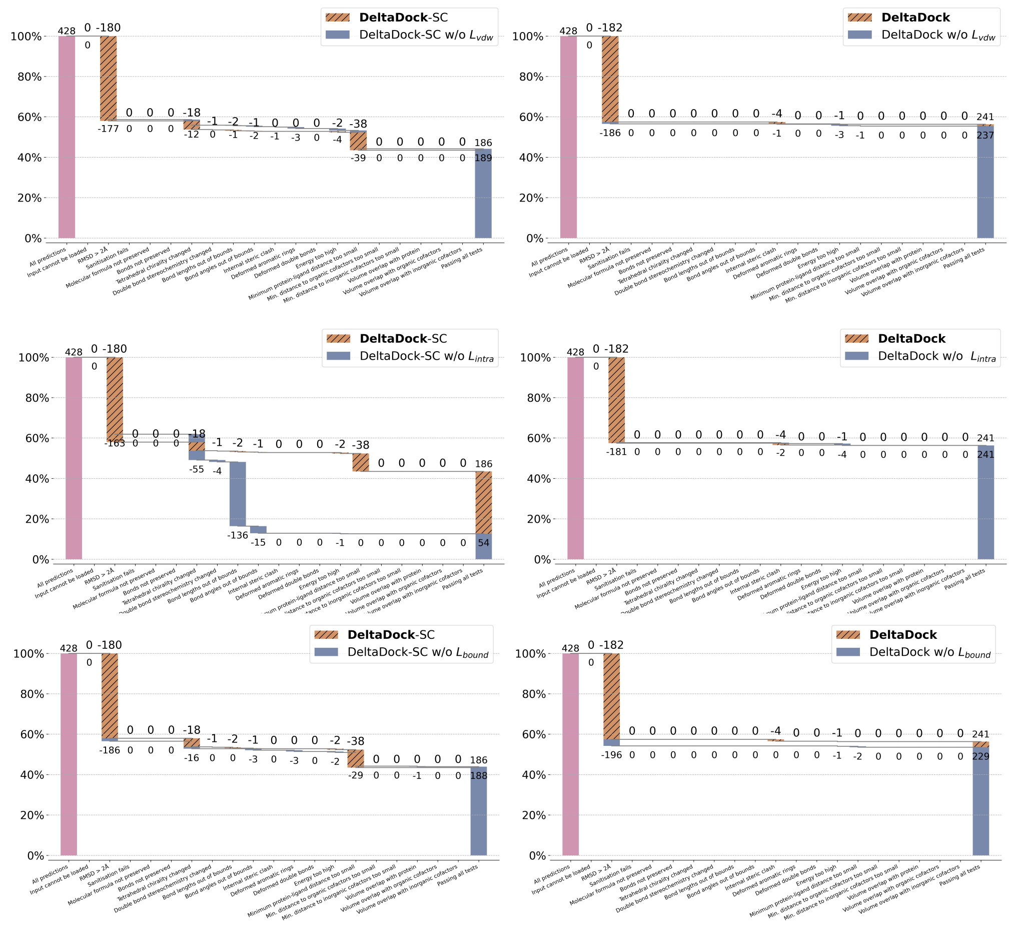

Figure 11: Site-specific docking performance on the PoseBusters dataset.

Table 11: Blind docking performance on the PDBbind dataset. The best results are **bold**, and the second best results are $\underline{\text{underlined}}$.

| Method | Time average | Ligand RMSD | | | | | | Centroid Distance | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Percentiles | | | | % below Threshold | | Percentiles | | | | % below Threshold | |
| | Seconds | 25% | 50% | 75% | mean | 2Å | 5Å | 25% | 50% | 75% | mean | 2Å | 5Å |
| QVINA-W | 49 | 2.5 | 7.7 | 23.7 | 13.6 | 20.9 | 40.2 | 0.9 | 3.7 | 22.9 | 11.9 | 41.0 | 54.6 |
| GNINA | 393 | 2.8 | 8.7 | 21.2 | 13.3 | 21.2 | 37.1 | 1.0 | 4.5 | 21.2 | 11.5 | 36.0 | 52.0 |
| VINA | 119 | 3.7 | 6.8 | 14.6 | 10.1 | 10.3 | 36.2 | 1.5 | 4.1 | 13.3 | 8.2 | 32.3 | 55.2 |
| SMINA | 146 | 3.8 | 8.1 | 13.5 | 12.1 | 13.5 | 33.9 | 1.3 | 3.7 | 16.2 | 9.8 | 38.0 | 55.9 |
| GLIDE | 1405 | 2.6 | 9.3 | 21.8 | 16.2 | 21.8 | 33.6 | 0.8 | 5.6 | 26.9 | 14.4 | 36.1 | 48.7 |
| DSDP | 1.22* | 1.0 | 3.0 | 7.9 | 7.2 | 42.4 | 59.8 | **0.3** | 1.0 | 4.8 | 4.9 | 60.3 | 75.8 |
| EquiBind | $\underline{0.03*}$ | 3.8 | 6.2 | 10.3 | 8.2 | 5.5 | 39.1 | 1.3 | 2.6 | 7.4 | 5.6 | 40.0 | 67.5 |
| TANKBind | 0.87* | 2.4 | 4.2 | 7.6 | 7.8 | 17.6 | 57.8 | 0.8 | 1.7 | 4.3 | 5.9 | 55.0 | 77.8 |
| DiffDock | 80* | 1.5 | 3.5 | 7.4 | 7.4 | 36.0 | 61.7 | 0.5 | 1.2 | 3.3 | 5.4 | 62.9 | 80.2 |
| FABind | 0.12* | 1.7 | 3.1 | 6.7 | 6.4 | 33.1 | 64.2 | 0.7 | 1.3 | 3.6 | 4.7 | 60.3 | 80.2 |
| FABind+ | 0.16* | 1.2 | 2.6 | $\underline{5.8}$ | $\underline{5.2}$ | 43.5 | $\underline{71.1}$ | $\underline{0.4}$ | 1.0 | 2.9 | **3.5** | 67.5 | $\underline{84.0}$ |
| FABind+(40) | 6.4* | 1.2 | 2.4 | **5.3** | **5.1** | 43.8 | **73.3** | 0.5 | 1.0 | **2.6** | **3.5** | $\underline{69.1}$ | **86.2** |
| **DeltaDock-SC** | 2.58* | $\underline{1.1}$ | $\underline{2.3}$ | 6.2 | 5.7 | **47.9** | 68.0 | $\underline{0.4}$ | **0.8** | **2.6** | 4.1 | **70.0** | 83.2 |
| **DeltaDock** | 2.97* | **1.0** | **2.2** | 6.4 | 5.8 | $\underline{47.4}$ | 66.9 | **0.3** | $\underline{0.9}$ | 2.9 | 4.2 | 66.7 | 83.2 |

[1] The time of consumption is denoted with * if it corresponds to GPU time; in the absence of this symbol, the time pertains to CPU time.

Table 12: Blind docking performance for PDBbind unseen dataset. The best results are **bold**, and the second best results are $\underline{\text{underlined}}$.

| Method | Ligand RMSD | | | | | | Centroid Distance | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Percentiles | | | | % below Threshold | | Percentiles | | | | % below Threshold | |
| | 25% | 50% | 75% | mean | 2Å | 5Å | 25% | 50% | 75% | mean | 2Å | 5Å |
| QVINA-W | 3.4 | 10.3 | 28.1 | 16.9 | 15.3 | 31.9 | 1.3 | 6.5 | 26.8 | 11.9 | 35.4 | 47.9 |
| GNINA | 4.5 | 13.4 | 27.8 | 16.7 | 13.9 | 27.8 | 2.0 | 10.1 | 27.0 | 15.1 | 25.7 | 39.5 |
| VINA | 5.0 | 9.6 | 19.0 | 12.8 | 7.8 | 25.5 | 2.2 | 6.1 | 17.8 | 10.9 | 24.1 | 41.8 |
| SMINA | 4.8 | 10.9 | 26.0 | 15.7 | 9.0 | 25.7 | 1.6 | 6.5 | 25.7 | 13.6 | 29.9 | 41.7 |
| GLIDE | 3.4 | 18.0 | 31.4 | 19.6 | 19.6 | 28.7 | 1.1 | 17.6 | 29.1 | 18.1 | 29.4 | 40.6 |
| DSDP | 1.0 | 4.5 | 10.0 | 9.1 | 37.2 | 54.9 | **0.2** | 1.5 | 6.5 | 7.5 | 54.2 | 69.0 |
| EQUIBIND | 5.9 | 9.1 | 14.3 | 11.3 | 0.7 | 18.8 | 2.6 | 6.3 | 12.9 | 8.9 | 16.7 | 43.8 |
| TANKBind | 3.0 | 4.9 | 8.7 | 9.3 | 4.9 | 52.1 | 1.3 | 2.3 | 4.4 | 7.2 | 41.5 | 76.1 |
| DiffDock | 2.9 | 6.4 | 16.3 | 12.0 | 17.2 | 42.3 | 1.0 | 2.7 | 14.2 | 9.8 | 43.3 | 62.6 |
| FABind | 2.2 | $\underline{3.4}$ | 8.3 | 7.7 | 19.4 | 60.4 | 0.9 | 1.5 | 4.7 | 5.9 | 57.6 | 75.6 |
| FABind+ | 1.6 | **3.3** | 8.9 | 7.0 | 34.7 | **63.2** | $\underline{0.5}$ | 1.5 | 4.2 | 5.1 | 58.3 | 77.1 |
| **DeltaDock-SC** | 1.2 | 3.9 | $\underline{7.8}$ | **5.7** | **40.8** | 60.5 | $\underline{0.5}$ | **1.2** | **4.0** | **4.1** | **65.5** | **78.9** |
| **DeltaDock** | **1.0** | 3.8 | **7.7** | $\underline{5.8}$ | **40.8** | $\underline{61.3}$ | **0.4** | **1.2** | $\underline{4.1}$ | $\underline{4.2}$ | $\underline{60.6}$ | **78.9** |

[1] The time of consumption is denoted with * if it corresponds to GPU time; in the absence of this symbol, the time pertains to CPU time.

### C.3.3 Detailed Tables of Performance on PDBbind

In the main manuscript, we have opted to present a concise table Table. 1 with key metrics due to the space constraint. Here, we present the detailed tables Table. 11 and Table. 12, which encompasses all metrics on PDBbind.

## D Broader Impacts and Limitations

### D.1 Broader Impacts

The development and maintenance of computational infrastructure for AI-assisted molecular docking represent a significant allocation of resources. Inefficient allocation or underutilization of these resources can potentially result in resource wastage.

### D.2 Limitations

One disappointing limitation is the reliance on external tools, such as SMINA for post-processing and the structure sampling algorithm for structure initialization. Although DeltaDock still achieves the best performance among GDL methods on the test unseen time split set without these tools, the overall performance degrades. Indeed, due to the limited training data, it's quite difficult to accomplish accurate, efficient, and physically reliable docking without any external tools. In the future, we will try to overcome this limitation by exploring pre-training strategies on large-scale datasets generated by docking methods or recently developed AlphaFold3 [50].

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction can accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations discussion can be found in Appendix. D.2.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide all details and parameters to reproduce our results, although the code and data are provided.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We will provide all data and code upon acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided all the training and test details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We follow the common practice of previous work, such as TANKbind [30] and FABind [11]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided sufficient information on the computer resources.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducts in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed it in Appendix. D.1

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper focuses on a drug discovery task, posing no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited datasets, code packages, and models properly in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.