What Ails Generative Structure-based Drug Design: Expressivity is Too Little or Too Much?

Rafał Karczewski
Aalto University

Samuel Kaski
Aalto University
University of Manchester

Markus Heinonen
Aalto University

Vikas Garg
Aalto University
YaiYai Ltd

Abstract

Several generative models with elaborate training and sampling procedures have been proposed to accelerate structure-based drug design (SBDD); however, their empirical performance turns out to be suboptimal. We seek to better understand this phenomenon from both theoretical and empirical perspectives. Since most of these models apply graph neural networks (GNNs), one may suspect that they inherit the representational limitations of GNNs. We analyze this aspect, establishing the first such results for protein-ligand complexes. A plausible counterview may attribute the underperformance of these models to their excessive parameterizations, inducing expressivity at the expense of generalization. We investigate this possibility with a simple metric-aware approach that learns an economical surrogate for affinity to infer an unlabelled molecular graph and optimizes for labels conditioned on this graph and molecular properties. The resulting model achieves state-of-the-art results using 100x fewer trainable parameters and affords up to 1000x speedup. Collectively, our findings underscore the need to reassess and redirect the existing paradigm and efforts for SBDD. Code is available at `https://github.com/rafalkarczewski/SimpleSBDD`.

1 INTRODUCTION

Identifying new molecules or *ligands* that bind well to a protein target and have desired properties is a key challenge of Structure-based drug design (SBDD). Since experimental determination of binding is expensive and time-consuming, deep generative models offer hope to accelerate SBDD by rapidly suggesting good candidate molecules for a given target protein, using experimental or *in silico* binding observations (Vamathevan et al., 2019; Bilodeau et al., 2022). Several approaches have been proposed, including autoregressive models (Peng et al., 2022; Luo et al., 2021a), variational autoencoders (Ragoza et al., 2022), reinforcement learning (Li et al., 2021), and diffusion models (Schneuing et al., 2024; Guan et al., 2023a,b; Le et al., 2024).

Despite a surge in interest from the machine learning community, these methods are observed to underperform empirically in terms of the docking scores, i.e., the estimated binding affinities of the candidates they generate. These findings are particularly surprising since many of the aforesaid methods jointly optimize for atom types and 3D coordinates (often using GNN-based architectures), with intricate training and sampling procedures. Since the success of SBDD hinges on identifying candidates that dock well (Cieplinski et al., 2023), understanding this phenomenon holds key to utilizing the promise of generative modeling in SBDD.

We take an initial but important step in this pursuit by proposing two contrasting hypotheses that shed light on different aspects of the problem. First, we investigate, theoretically, the possibility that the shortcomings of message-passing GNNs (Scarselli et al., 2009; Joshi et al., 2023), e.g., due to their inability to compute graph properties (Garg et al., 2020), might propagate to the protein-ligand complexes, yielding insufficient representations. It turns out that we can expose the limits on the expressivity of GNNs to distinguish ligands conditioned on the same protein target. To our knowledge, these are the first such results in conditional drug discovery contexts, motivating the design and analysis of more effective models that can mitigate the representational limitations.

Second, we also study and analyze, empirically, a counterview that the generative models for SBDD might

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

already be getting too sophisticated, owing, e.g., to their excessive parameterizations, and simpler methods may provide strong baselines. Some preliminary evidence on the efficacy of simple baselines for the unconditional molecule generation setting, but not SBDD, has emerged recently (Tripp and Hernández-Lobato, 2023) stemming from similar concerns.

Concretely, we identify a key issue that has eluded attention heretofore; namely, that the existing methods predominantly focus on obtaining rich representations for drug properties such as QED with bulky models, while being oblivious to binding affinity as they seek to match the observed ligand distribution (which often includes molecules with suboptimal affinities). We address these issues with a novel simple two-phase generative method *SimpleSBDD* that draws inspiration from the success of performance-aware methods in other domains (Joachims, 2005), and optimizes for both the metrics of interest, i.e. molecular properties and estimated binding affinity.

Specifically, we first learn an economical affinity surrogate to infer an unlabelled molecular graph, and then optimize its atom labels and coordinates for other properties. Albeit segregating optimization over structure from labels loses some information, we observe that our approach already achieves state of the art in SBDD with up to 1000x speed up and 100x fewer parameters than prior methods. Importantly, we represent the unlabelled molecular graphs with features that are unlearnable by GNNs.

Note that such massive reduction in computation and model size bears significant potential benefits for the docking software in particular, and the overall SBDD pipeline in general. Streamlining the generation of *hits*, i.e., promising initial candidates that bind well to a specified protein target would afford re-channeling of limited resources for *lead* optimization as well as more stringent and precise validation stages than docking, namely, design and analysis of simulations for molecular dynamics as well as wet lab experiments.

1.1 Contributions of this work

We summarize our contributions below. This work

1. **(Topical)** draws attention to the disparity between the promise and the observed empirical performance of generative models for SBDD, outlining concerns pertaining to their expressivity;

2. **(Theoretical)** initiates a formal analysis for limits on the expressivity of methods for SBDD, exposing the representational challenges inherent in GNN models for encoding protein-ligand complexes;

3. **(Methodological)** introduces performance-aware optimization in SBDD settings, using a simple generative model that optimizes for both docking scores and physicochemical properties; and

4. **(Empirical)** demonstrates the versatility of the proposed approach SimpleSBDD using three instantiations tailored to problems of drug repurposing (suggesting candidates from a known ligand set), generation of novel ligands, and property optimization that seeks to control both molecular properties and binding. SimpleSBDD outperforms existing methods with orders of magnitude fewer parameters and faster runtime.

2 PRELIMINARIES AND RELATED WORK

Deep (Geometric) Learning for Drug Design. Deep learning has enabled progress on various facets of drug design across molecular generation (Jin et al., 2018; Zang and Wang, 2020; Verma et al., 2022; Shi et al., 2020; Luo et al., 2021b; Hoogeboom et al., 2022; Igashov et al., 2024), molecular optimization (Jin et al., 2019) and drug repurposing (Pushpakom et al., 2019). It has also accelerated advancements in protein folding and design (Jumper et al., 2021; Wu et al., 2022; Ahdritz et al., 2024; Verkuil et al., 2022; Hie et al., 2022; Shi et al., 2023; Watson et al., 2022; Ingraham et al., 2023; Wu et al., 2024; Verma et al., 2023) and docking (Stärk et al., 2022b; Ganea et al., 2022; Corso et al., 2023). We refer the reader to the surveys by Chen et al. (2018) and Pandey et al. (2022).

Graph Neural Networks (GNNs) have emerged as a workhorse for modeling graph data, such as molecules (Scarselli et al., 2009; Kipf and Welling, 2017; Hamilton et al., 2017; Veličković et al., 2018). They can be extended to encode important symmetries in geometric graphs (Satorras et al., 2021), so have found success across diverse tasks in the drug design pipeline including docking (Corso et al., 2023) and molecular property prediction (Stärk et al., 2022a).

However, GNNs have known representational limits (Xu et al., 2019; Morris et al., 2019; Maron et al., 2019; Garg et al., 2020; Wang and Zhang, 2022; Puny et al., 2023a; Joshi et al., 2023). There have been improvements proposed (Bouritsas et al., 2022; Vignac et al., 2020; Puny et al., 2023b) albeit potentially at the expense of generalization (Karczewski et al., 2024).

Molecular properties The goal of SBDD is to generate drug candidates with high binding affinity to a target protein, while controlling for other properties. We list below common molecular properties, calculated

in practice using the RDKit software (Landrum et al., 2020), that are relevant to drug design (Bilodeau et al., 2022; Peng et al., 2022): 1) Binding affinity – the strength of the connection between the ligand and the protein. A common surrogate for affinity is the Vina score (Alhossary et al., 2015); 2) LogP – logarithm of the partition coefficient, which is a solubility indicator; 3) QED – quantitative estimate of drug-likeness, a mixture of properties correlated with drugs; 4) SA – Synthetic accessibility, an estimate of how easy it would be to synthesize the molecule; and 5) Lipinski's Rule of Five – A heuristic about whether a molecule would be an active oral drug.

Setting. We view proteins as labelled graphs: each pocket $G = (V, E)$ consists of nodes $V = \{v_1, \ldots, v_N\}$ with v_i representing atoms, and edges $E \subseteq V \times V$. Each node $v = (a, \mathbf{s})$ is associated with an atom type $a \in \mathcal{A} = \{\mathtt{C}, \mathtt{N}, \mathtt{O}, \ldots\}$ and 3D coordinates $\mathbf{s} \in \mathbb{R}^3$. We include superscripts G^P and G^M to distinguish between protein and molecule graphs. We also distinguish between a labelled molecular graph G^M and unlabelled one U^M. The unlabelled graph contains information about the number of nodes and edges between them, but no information about atom types or 3D coordinates.

The problem of SBDD can be posed as modeling the conditional distribution $p(G^M|G^P)$. The challenge is to generate candidate ligands with high binding affinity to a target protein. We begin our analysis with a theoretical examination of expressivity limits of GNNs in the SBDD context.

3 REPRESENTATION LIMITS OF GNNS FOR SBDD

The generative models for SBDD are quickly increasing in size (see Table 1) aiming to expand their expressivity. However, we bring attention to the representational issues concerning large models. Specifically, we show that there exist molecular graphs that cannot be distinguished by GNN models regardless of their depth, even in the additional protein context. Consequently, certain graph properties, including the number of cycles, diameter, and sizes of cycles, cannot be learned by message-passing graph neural networks (MPGNNs). Interestingly, as we will demonstrate in Section 4, these features are very useful in predicting binding affinity.

It is known that there exist distinct molecular structures that cannot be distinguished using message-passing graph neural networks (MPGNNs) (Sato, 2020; Garg et al., 2020). Following the notation of Garg et al. (2020), we consider *Locally Unordered* GNNs (LU-GNNs), summarized as follows. The updated embedding $h_v^{(l)}$ of node v at layer l is defined as

$$\begin{aligned} m_{u\to v}^{(l-1)} &= \phi(h_u^{(l-1)}, e_{uv}) \\ \tilde{h}_v^{(l-1)} &= \mathrm{AGG}\{m_{u\to v}^{(l-1)} \mid u \in N(v)\} \\ h_v^{(l)} &= \mathrm{COMBINE}\{h_v^{(l-1)}, \tilde{h}_v^{(l-1)}\}, \end{aligned}$$

where $N(v)$ denotes the set of neighbours of v, e_{uv} are edge features and ϕ is any function. It is known that there exist non-isomorphic graphs, for which LU-GNNs produce identical embeddings (Garg et al., 2020). We now note that this result also holds for graphs embedded in 3D space. Let us introduce *Locally Unordered 3D* GNNs (LU3D-GNN) which aggregate embeddings for each node based on the embeddings of the neighbours, as well as their distance to these neighbours. Formally, in LU3D-GNNs, the messages are defined as:

$$m_{u\to v}^{(l-1)} = \phi\left(h_u^{(l-1)}, e_{uv}, \|x_u - x_v\|\right) \tag{1}$$

where x_v is the 3D position of v. We have the following result.

Lemma 3.1. *There exist connected non-isomorphic geometric graphs that differ in the number of conjoined cycles, girth, size of the largest cycle and cut-edges that LU3D-GNNs cannot distinguish.*

On the lefthand side of Figure 1, we show two graphs that are not isomorphic and differing in all mentioned properties, but which cannot be distinguished by LU3D-GNNs. For clarity of presentation, we presented the graphs in 2D, but any 3D configuration having all edges of equal length can be used. We prove that they cannot be distinguished by LU3D-GNNs in Appendix K.

We extend this result to the SBDD context. Specifically, we analyze pairs of graphs corresponding to protein-ligand complexes. Let $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ be any two graphs and $\mathcal{C}(G_1, G_2)$ denote a *complex* graph, i.e. $\mathcal{C}(G_1, G_2) = (V, E)$, where

$$V = V_1 \cup V_2 \text{ and } E = E_1 \cup E_2 \cup V_1 \times V_2 \tag{2}$$

and, importantly, features for all added edged $e_{uv} \in V_1 \times V_2$ only depend on the features of nodes at their endpoints. We have the following results.

Proposition 3.2.

(i) If G_1 and G_2 are indistinguishable for LU-GNNs, then for any graph P, the complex graphs $\mathcal{C}(P, G_1)$, $\mathcal{C}(P, G_2)$ are also indistinguishable for LU-GNNs.

(ii) There exist ligand graphs G_1, G_2 differing in graph properties listed in Lemma 3.1, such that for any protein P, the complexes $\mathcal{C}(P, G_1)$, $\mathcal{C}(P, G_2)$ are indistinguishable for models using LU3D-GNNs

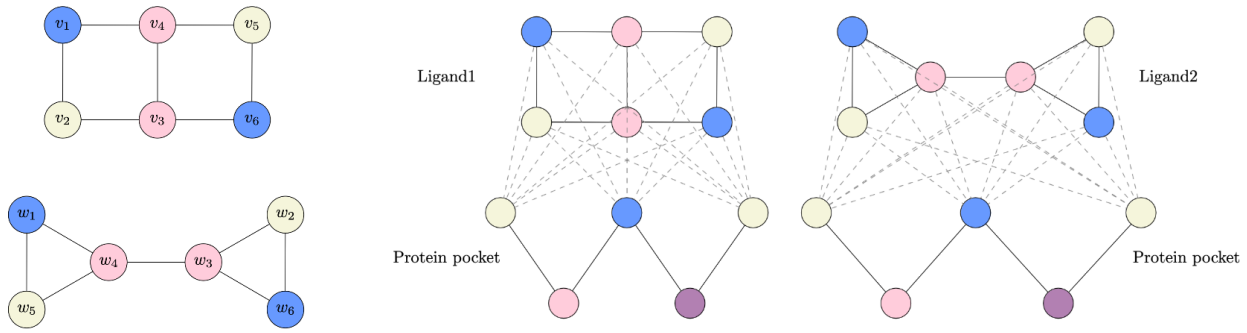


Figure 1: **Some ligands cannot be distinguished by GNNs even with additional protein context** Left: Construction for Lemma 3.1; Two non-isomorphic graphs differing in all properties stated in Lemma 3.1, but for which LU-GNNs produce identical embeddings. Right: Complex graphs constructed by joining the ligand graphs with the same protein graph remain identical whenever ligand graphs cannot be differentiated.

or LU-GNNs for intra-ligand and intra-protein message passing and LU-GNNs for inter ligand-protein message passing.

The righthand side of Figure 1 visualizes the proposed statement. The proof can be found in Appendix K. Proposition 3.2 shows that the representational issues of GNNs carry over to the context of SBDD. Specifically, this shows that there exist distinct protein-ligand structures that cannot be distinguished with LU-GNN models regardless of their depth or number of parameters. Importantly, LU-GNNs' inability to distinguish graphs that differ in certain properties implies that LU-GNNs are unable to learn these properties.

We now take a different perspective and show that generative models for SBDD can be significantly reduced in size and computational complexity without sacrificing performance.

4 SIMPLE STRUCTURE-BASED DRUG DESIGN

In this section, we propose a significantly simplified generative model for SBDD with two crucial components: innovative model decomposition and explicit incorporation of estimated binding optimization within the model. This differs from the prevailing paradigm that primarily focuses on learning the data distribution.

4.1 Decoupling the unlabelled molecular graph from atom types

One of the main innovations underlying our framework is the separation of molecular representation into the unlabelled molecular graph and the labels, the atom types. Our motivation to adopt such an approach is primarily rooted in empirical findings that highlight the strong relationship between the structure of molecules and their predicted binding affinities, as approximated by the Vina software. In a preliminary experiment, we investigated how Vina binding scores behave when we modify the atom types and coordinates while keeping the underlying graph intact.

Remarkably, our analysis showed a high correlation at $\rho = 0.83$, between the scores of the original molecules and their modified counterparts. This result highlights that a considerable portion of the variability in predicted binding affinity can be attributed to the information contained within the unlabelled molecular graph itself, even before incorporating atom-specific details (details in Appendix A). Note that features characterizing the unlabeled molecular graph, such as the diameter and the number of rings, are unlearnable by LU-GNNs (Section 3). We will leverage this insight in our model.

It is important to recognize that these findings pertain to the Vina software and not the biological process of binding. While Vina is the most prevalent approximation, it remains just that – an approximation. Nonetheless, upon conducting similar experiments with Gnina (McNutt et al., 2021), a reportedly significantly more accurate binding approximation than Vina, we found similar results (Appendix A.3). This suggests that the observed behavior is not solely an artifact of Vina but may be more widespread.

Inspired by these findings, we propose a new model SimpleSBDD (Figure 2). We first generate the high-binding unlabelled graph structures U^M, and generate atom types $\mathbf{a}^M$ independently of the protein. The model is defined to approximate the conditional distribution

$$p(G^M|G^P) \approx p(\mathbf{a}^M, \mathbf{s}^M|U^M, \cancel{G^P})p(U^M|G^P), \quad (3)$$

where we assume atoms $\mathbf{a}^M$ are conditionally independent of the protein pocket G^P given the unlabelled graph structure U^M. This approximation does incur

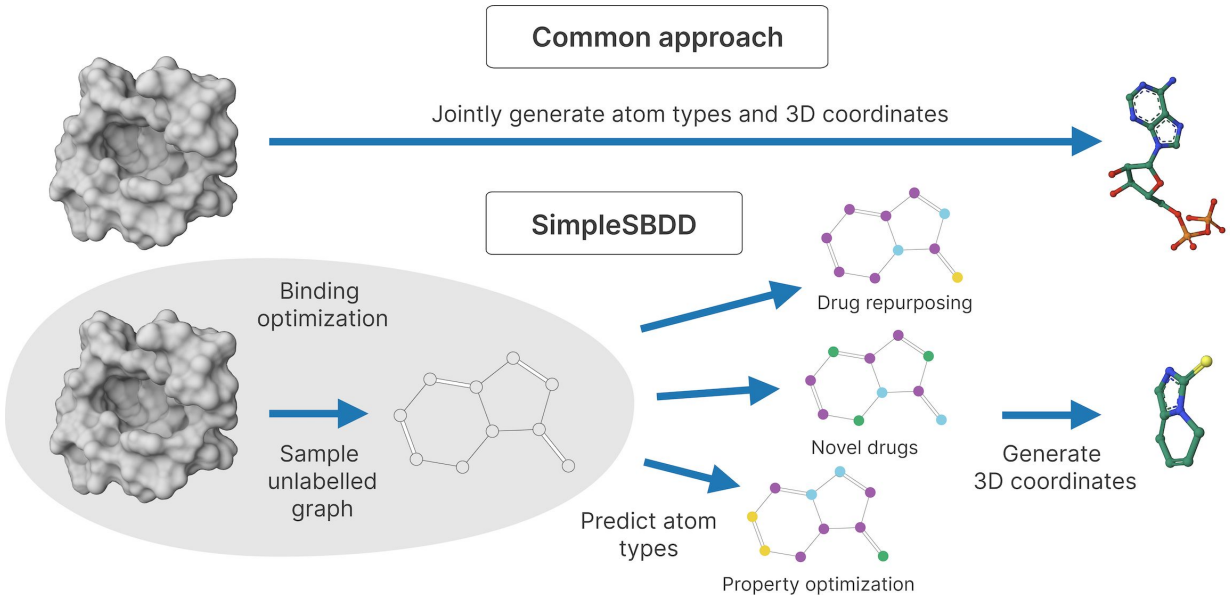


Figure 2: **Comparison of SimpleSBDD to common approaches**. Top: SBDD approaches commonly learn to approximate the data distribution of atom types and 3D coordinates conditioned on the protein pocket. Bottom: SimpleSBDD first generates the unlabelled graph explicitly optimized for estimated binding affinity. Then it predicts atom types using different strategies designed for solving different tasks independently of the protein pocket. Finally, it generates a 3D configuration.

information loss as expected. However, as we will see in the next section, performance-aware learning mitigates this effect. The generative model thus reduces to two independent components: unlabelled graph sampler and atom sampler. We discuss them below.

4.2 Performance-aware learning

Typically, the unlabelled graph model $p(U^M|G^P)$ is trained to match the data distribution. However, we hypothesize that observed ligand-protein complexes contain examples with suboptimal binding, and propose to focus on the most promising complexes with the following procedure. We first learn a scoring model g_θ to predict binding affinity from unlabelled graph alone,

$$g_\theta(U^M, G^P) \approx \text{Vina}(G^M, G^P). \tag{4}$$

To represent U^M we use properties describing its structure such as its number of rings, number of rotatable bonds and graph diameter, which cannot be learned by MPGNNs (See Appendix D for details). Next, we use g_θ to score unlabelled graphs from a repository of molecules, and choose those with the best predicted affinities. We define a generative distribution

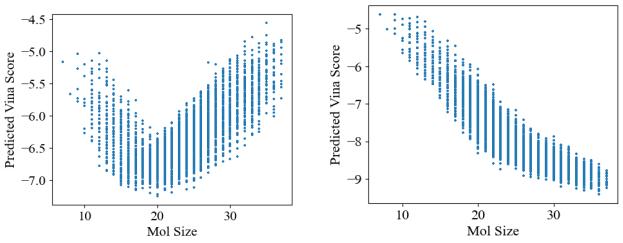
$$\begin{aligned} &p_\theta(U^M|G^P) = \text{UNIFORM}(U(G^P)), \\ &U(G^P) = \left\{U^M \in \mathcal{U}^M \mid g_\theta(U^M, G^P) \in [v_{\min}, v_{\max}]\right\}, \end{aligned} \tag{5}$$

where $\mathcal{U}^M$ is a database of unlabelled graph structures, and $v_{\min}, v_{\max}$ are threshold hyperparameters. We set $v_{\min}$ and $v_{\max}$ to be the 5th and 10th percentiles of all predictions for $\mathcal{U}^M$ to avoid outliers (Appendix B). As $\mathcal{U}^M$, we used the ZINC250k (Irwin et al., 2012) instead of the CrossDocked2020 (Francoeur et al., 2020), which only contains 8K unique ligands.

We note that instead of sampling from a database, one can define a generative model over unlabelled graphs. We have not investigated this due to substantial molecular variability even with predefined unlabelled graphs. Specifically, one can generate at least a thousand distinct and chemically diverse molecules sharing the same unlabelled graph (Appendix C).

Training data for g_θ To train the scoring model g_θ, we need pairs of protein-ligand pairs with their affinity estimates. We chose not to use the CrossDocked2020 dataset, the gold standard for SBDD, for the following reasons: 1) even though it contains 100K protein-ligand pairs, there are only around 8K unique ligands, 2) for a given protein pocket there may exist ligands with better affinity than what is present in the data and 3) for the scoring model to assign good values only to good structures, it needs to be trained also on examples with very poor binding affinity which are not available.

To address the above, we constructed a dataset ourselves. We sampled 1000 diverse ligand-protein com-



(a) Optimal ligand size ≈ 20 (b) Optimal ligand size ≥ 38

Figure 3: **Optimal ligand size has two modes**. Representative examples showing that the scoring model learns the optimal ligand size which is different for different proteins.

plexes from CrossDocked2020, and included 50 additional random molecules from the ZINC250k. This resulted in 51,000 pairs, for which we computed the Vina scores. The scoring model g_θ was trained with stochastic gradient descent to minimize the mean squared error between prediction g and Vina scores. See Appendix D.1 for details.

Scoring model detects optimal molecule size We visualize the predictions of g as a function of the size of the molecule. We notice that protein pockets fall under two cases: either there is an optimal size of the ligand (Figure 3a) or there is a monotonic relationship favouring larger ligands (Figure 3b). This is because the largest available molecule is 38 atoms, which is smaller than the learned optimum.

4.3 Atom sampler

The atom sampler $p(\mathbf{a}^M, \mathbf{s}^M | U^M)$ generates atom types and coordinates of each atom. As mentioned in Section 4.1, the precise 3D conformation has marginal effect on the Vina score (due to redocking), and therefore we simply generate any valid 3D configuration with RDKit. We choose a factorisation

$$p(\mathbf{s}^M, \mathbf{a}^M | U^M) = p(\mathbf{s}^M | \mathbf{a}^M, U^M) p(\mathbf{a}^M | U^M), \quad (6)$$

where $p(\mathbf{s}^M | \mathbf{a}^M, U^M)$ is an off-the-shelf conformation generation model available in RDKit, which we discuss in more detail in Appendix E. In practice, the 3D configuration of the ligand must be in the vicinity of the protein pocket in order to correctly compute its binding affinity using Vina. We therefore train a separate model which predicts the center of mass of the ligand based on the protein pocket, and use it to center the conformers. See Appendix F for more details.

To define the generative model over the atom types given unlabelled structure, we adapt an existing generative model MoFlow (Zang and Wang, 2020), consisting of an unlabelled graph model f_U and a conditional atom model $f_{\mathbf{a}|U}$. The atom model induces a conditional distribution $p_{f_{\mathbf{a}|U}}(\mathbf{a}^M | U^M)$, which we use as our atom sampling distribution:

$$p(\mathbf{a}^M | U^M) \coloneqq p_{f_{\mathbf{a}|U}}(\mathbf{a}^M | U^M). \quad (7)$$

We use a pre-trained MoFlow, so omit it from the trainable parameter count (Table 1).

5 RESULTS

Data We follow Peng et al. (2022); Schneuing et al. (2024) and use the CrossDocked2020 dataset (Francoeur et al., 2020) for evaluating the models. We use the same train-test split, which separates protein pockets based on their sequence similarity computed with MMseqs2 (Steinegger and Söding, 2017) and contains 100,000 train and 100 test protein-ligand pairs.

5.1 Assessing general properties of sampled molecules

We follow the evaluation procedure described by Peng et al. (2022). For each of 100 test protein pockets, we sample 100 molecules. As evaluation metrics we report 1) **Vina Score** estimating binding affinity (Alhossary et al., 2015), 2) **High Affinity**, which is the percentage of generated molecules with binding affinity at least as good as ground truth, 3) **QED**, a quantitative estimate of druglikeness, 4) **SA**, synthetic accesibility, 5) **Diversity** defined by average pairwise Tanimoto dissimilarity for the generated molecules in the pocket, 6) **Novelty**, defined by average Tanimoto dissimilarity to the most similar molecule from training set, 7) **#Params**, the number of trainable parameters and 8) **Time** to generate 100 molecules in seconds. We include more molecular metrics in Appendix G. As baselines, we use eight recent methods for SBDD: Pocket2Mol (Peng et al., 2022), DiffSBDD (Schneuing et al., 2024), TargetDiff (Guan et al., 2023a), DecompDiff (Guan et al., 2023b), FLAG (Zhang et al., 2022), DrugGPS (Zhang and Liu, 2023), D3FG (Lin et al., 2024) and EQGAT-diff (Le et al., 2024).

We report the results in Table 1. Clearly, SimpleSBDD significantly outperforms all baselines in terms of estimated binding affinity while being up to 1000x faster despite being run solely on CPU (all baselines use GPU for sampling). Additionally, our method has only 23k trainable parameters, which is two orders of magnitude less than the baselines. We provide more details on the baseline comparison in Appendix H. We also provide a visualization of our model predictions in Figure 4.

Table 1: **SimpleSBDD finds high-quality drug candidates up to 1000x faster than competing methods**. Properties of CrossDocked2020 data and generation methods. † denotes method run exclusively on CPU. "-" denotes that a metric was not reported and we were unable to estimate. Error bars correspond to the standard deviation across test protein pockets. See Appendix H for comparison details.

	Vina Score (kcal/mol, ↓)	High Affinity (↑)	QED (↑)	SA (↑)	Diversity (↑)	Novelty (↑)	#Params (↓)	Time (s, ↓)
Test set	-6.99 ± 2.16	-	0.48 ± 0.21	0.73 ± 0.14	-	-	-	-
DiffSBDD (2024)	-6.29 ± 1.93	0.37 ± 0.31	0.49 ± 0.19	0.63 ± 0.14	**0.79** ± 0.07	**0.54** ± 0.14	3.5M	135 ± 52
Pocket2Mol (2022)	-7.10 ± 2.56	0.55 ± 0.31	0.57 ± 0.16	0.74 ± 0.13	0.72 ± 0.16	0.45 ± 0.16	3.7M	2504 ± 220
FLAG (2022)	-7.25 ± 2.25	0.58 ± 0.24	0.50 ± 0.17	0.75 ± 0.16	0.70 ± 0.15	0.44 ± 0.17	11M	1048 ± 682
DrugGPS (2023)	-7.28 ± 2.14	0.57 ± 0.23	**0.61** ± 0.22	0.74 ± 0.18	0.68 ± 0.15	0.47 ± 0.15	14.7M	1008 ± 554
TargetDiff (2023a)	-6.91 ± 2.25	0.52 ± 0.32	0.48 ± 0.20	0.58 ± 0.13	0.72 ± 0.09	0.47 ± 0.14	2.5M	3428 ± NA
DecompDiff (2023b)	-6.76 ± 1.64	0.46 ± 0.36	0.45 ± 0.21	0.61 ± 0.14	0.68 ± 0.10	0.52 ± 0.13	5.0M	6189 ± NA
D3FG (2024)	-6.96 ± NA	0.46 ± NA	0.50 ± NA	**0.84** ± NA	-	-	-	-
EQGAT-diff (2024)	-7.42 ± 2.33	-	0.52 ± 0.18	0.70 ± 0.20	0.74 ± 0.07	-	12.3M	-
SimpleSBDD (Ours)	**-7.78** ± 1.47	**0.71** ± 0.34	**0.61** ± 0.18	0.69 ± 0.09	0.68 ± 0.06	0.51 ± 0.10	**23K**	**3.9**† ± 0.9

5.2 Additional evaluation

We have demonstrated that SimpleSBDD is very competitive on the standard evaluation criteria for SBDD models. We now discuss its broader applications.

Property optimization Due to our novel decomposition of the model into the unlabelled graph and atom labels, we can optimize the estimated binding affinity and other properties simultaneously. This is especially important, since molecules without desirable drug-like properties are of diminished practical utility even if the predicted binding is strong. Specifically, given the target protein we first generate an unlabelled graph with strong predicted binding affinity. As we noted in Section 4.2 and elaborated on in Appendix C, there is large diversity in the chemical properties of molecules that share the unlabelled graph. Therefore, for the sampled unlabelled graph, there is enough flexibility in atom types to enable molecular property optimization.

As an illustrative example, we follow Gómez-Bombarelli et al. (2018) and choose $5 \cdot \text{QED} + \text{SA}$ as the property mixture to optimize alongside predicted binding affinity. Concretely, 50 proposals are sampled using the atom type model defined in Equation (7) and the candidate with the best property values is selected. We denote this model variant SimpleSBDD–$\mathcal{PO}$. With this additional procedure we can significantly improve all molecular metrics without sacrificing binding affinity whilst remaining an order of magnitude faster than baselines. See Appendix I for more details.

Comparison with optimization-based methods So far we have compared our method with multiple baselines which generate molecules conditioned on a target protein. Neither SimpleSBDD nor any of these baselines has access to the Vina software during generation. We now also compare with *optimization-based* methods, i.e. algorithms, which generate molecules by explicitly using Vina during sampling to directly optimize it. Specifically, we compare SimpleSBDD–$\mathcal{PO}$ with the three best performing methods, i.e. Reinforced Genetic Algorithm (Fu et al., 2022), 3D-Monte Carlo Tree Search (3D-MCTS) (Du et al., 2023) and AutoGrow4.0 (Spiegel and Durrant, 2020). We present the results in Table 2.

It is evident that SimpleSBDD–$\mathcal{PO}$ is significantly more efficient than all the other methods. Note that these baselines are not only the best performing, but also the fastest according to (Fu et al., 2022). What makes them much slower than SimpleSBDD are the repeated evaluations of the Vina software, which is approximately 20000x slower than e.g. computing QED (20s vs 0.001s). Furthermore, the only method outperforming SimpleSBDD in terms of Vina score, AutoGrow4, is significantly worse in terms of QED. SimpleSBDD offers an attractive tradeoff between predicted binding affinity and the predicted quality of the generated drug candidates at a much lower computational cost.

Efficient scoring for drug repurposing SBDD is a problem of generating novel drugs likely to bind to a given target pocket. However, an equally important application is *drug repurposing*, a problem of investigating new uses for existing molecules, which is closely related to determining whether a drug can effectively bind to a specific protein target. This is a significant application since, unlike samples generated by machine learning models, existing drugs have experimentally verified properties and their synthesizability is established.

We can use our scoring model (Equation (4)) to scan databases of existing drugs and select ones with the best predicted binding. Moreover, our implementation of the scoring model allows us to do that very efficiently due to the fact that protein embedding needs only be computed once when scoring multiple molecules. Specifically, it allows for *scoring 9100 molecules per second on a single CPU*.

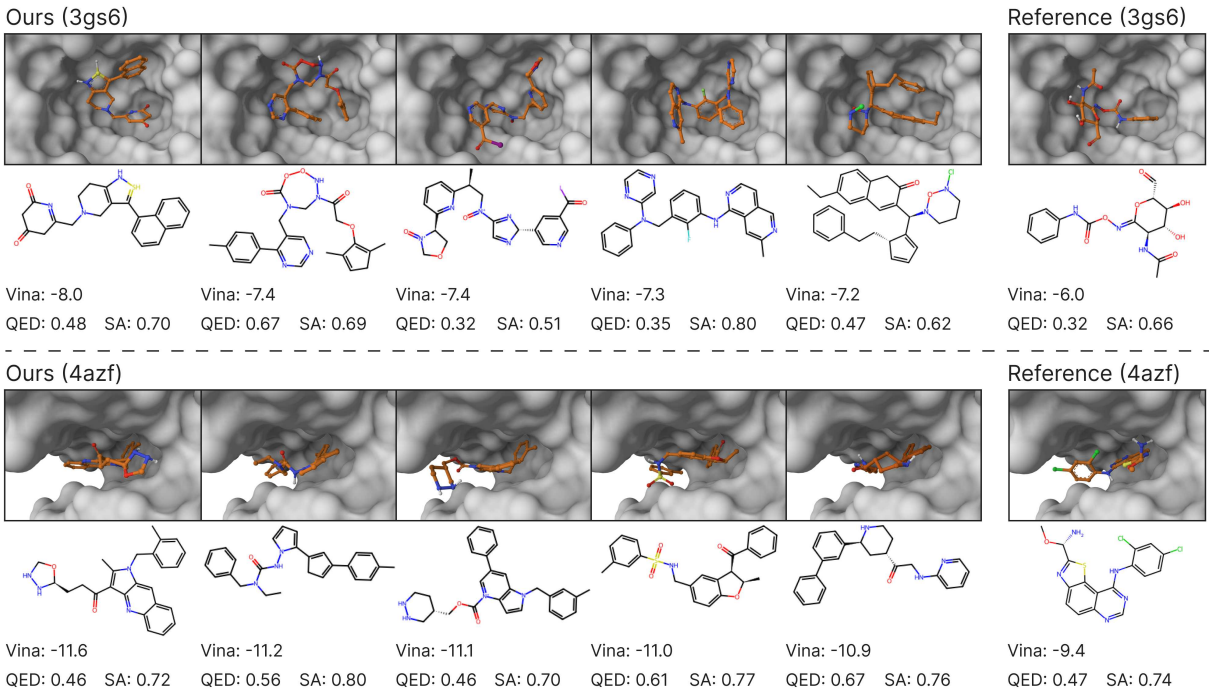


Figure 4: **SimpleSBDD generates diverse drug candidates with stronger predicted binding than reference molecules.** We visualize the predictions of our model for two randomly chosen proteins (PDB ids 3gs6 and 4azf). We choose 5 molecules with best predicted binding affinities for each protein and compare their Vina score, QED and synthetic accessibility with the reference molecule.

Table 2: **SimpleSBDD is significantly faster than optimization-based methods**. Comparison with optimization-based methods. † denotes method run exclusively on CPU. Error bars correspond to the standard deviation across test protein pockets.

	Vina Score (kcal/mol, ↓)	High Affinity (↑)	QED (↑)	SA (↑)	Diversity (↑)	#Params (↓)	Time (s, ↓)
Test set	-6.99 ± 2.16	-	0.48 ± 0.21	0.73 ± 0.14	-	-	-
RGA (2022)	-6.93 ± 1.17	0.53 ± 0.41	0.46 ± 0.15	**0.80** ± 0.07	**0.76** ± 0.01	341K	11576 ± 3717
3D-MCTS (2023)	-7.55 ± 1.32	0.66 ± 0.38	0.65 ± 0.14	0.78 ± 0.07	0.62 ± 0.07	**0**	4150 ± 313
AutoGrow4 (2020)	**-8.33** ± 1.55	**0.81** ± 0.28	0.36 ± 0.17	0.67 ± 0.10	0.65 ± 0.06	**0**	10800† ± 0
SimpleSBDD–$\mathcal{PO}$	-7.98 ± 1.46	0.75 ± 0.35	**0.80** ± 0.10	0.73 ± 0.08	0.67 ± 0.06	23K	**115**† ± 11

To showcase the practicality and efficiency of our scoring model in the context of drug repurposing, we perform the following experiment. For each protein pocket, we randomly sample 16384 molecules from a database and select 100 of them with predicted Vina scores between 5th and 10th percentile. We then compare the average binding affinity and other molecular properties with the reference molecules of the target protein. We found that for each protein pocket it takes around 2 seconds to find 100 diverse molecules with much higher binding affinity and improved other molecular metrics. Furthermore, we repeated the experiment with ChEMBL dataset (Mendez et al., 2018; Davies et al., 2015) and found that g_θ generalizes well across molecular databases. See Appendix J for details.

6 CONCLUSION AND BROADER IMPACT

We advocate a shift in focus from mere model expressivity to robust generalization. Specifically, we bring to the forefront inherent limitations in expressivity of GNNs and prove that they carry over to the multibody domain of SBDD. Furthermore, we show that with a novel model decomposition and explicit binding optimization built into the model, we can outperform state of the art with significantly simplified models. The unprecedented efficiency and success of SimpleSBDD highlights the potential to reshape the approach to SBDD but also paves the way for streamlining docking software, especially when assessed with widely adopted

tools like Vina and Gnina.

However, there are limitations to our findings. In particular, while we rely on docking software such as Vina, the gold standard for evaluation in the SBDD realm, the ideal validation would entail wet lab experiments or molecular dynamics simulations. That said, such methods are often prohibitively expensive and could utilize the significant resources afforded by models such as SimpleSBDD.

The very nature of SimpleSBDD as a generative model, especially with its property optimization capabilities, brings with it risks. As SBDD models improve at designing molecules with specific properties, we must remain vigilant of the potential unintended outcomes, since streamlining the design process could accelerate the design of harmful biochemicals.

7 LIMITATIONS

We acknowledge limitations of current docking-based approaches in real-world drug discovery. While docking scores are a widely used metric in SBDD, they should be interpreted with caution due to the following concerns:

Docking Score Reliability: A high docking score does not necessarily guarantee success in subsequent molecular simulations or experimental validations. Docking scores indicate the predicted binding pose of a ligand to a protein target and should not be overinterpreted as definitive indicators of biological activity.

Dataset Curation and Bias: The quality and curation of datasets can significantly impact docking outcomes. Protein structures from the Protein Data Bank (PDB) often exist as complexes with specific ligands. It is not uncommon to delete the ligand from the complex and dock new molecules instead. This can influence the docking results.

Biological Context: Docking scores do not distinguish between different functional outcomes of binding, such as whether a molecule acts as an inhibitor or an agonist. This limits their utility in predicting the nuanced biological effects of candidate molecules

Correlation with Experimental Data: Docking scores and experimental binding affinities are not always strongly correlated, and docking methods may lack proper calibration against real-world biochemical and biophysical data.

Binding Site Challenges: The binding site of a protein target might be suboptimal or unsuitable for a candidate ligand, further complicating the interpretation of docking results.

Such limitations underscore the need to treat docking scores with caution, and provide motivation for our work – we urge the community to critically evaluate the existing paradigm and redirect the efforts for SBDD (saving the computation expense and time using accelerated approaches such as the one we advocate, and paying additional attention to more reliable investigations such as molecular simulations).

Acknowledgments

This work was supported by the Finnish Center for Artificial Intelligence (FCAI) under Flagship R5 (award 15011052). VG also acknowledges the support from Saab-WASP (grant 411025), Academy of Finland (grant 342077), and the Jane and Aatos Erkko Foundation (grant 7001703). RK thanks Paulina Karczewska for her help with preparing figures.

References

Ahdritz, G., Bouatta, N., Floristean, C., Kadyan, S., Xia, Q., Gerecke, W., O'Donnell, T. J., Berenberg, D., Fisk, I., Zanichelli, N., et al. (2024). Openfold: Retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization. *Nature Methods*.

Alhossary, A., Handoko, S. D., Mu, Y., and Kwoh, C.-K. (2015). Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics*.

Bilodeau, C., Jin, W., Jaakkola, T., Barzilay, R., and Jensen, K. F. (2022). Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. (2022). Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., and Blaschke, T. (2018). The rise of deep learning in drug discovery. *Drug discovery today*.

Cieplinski, T., Danel, T., Podlewska, S., and Jastrzebski, S. (2023). Generative models should at least be able to design molecules that dock well: A new benchmark. *Journal of Chemical Information and Modeling*.

Corso, G., Stärk, H., Jing, B., Barzilay, R., and Jaakkola, T. S. (2023). Diffdock: Diffusion steps, twists, and turns for molecular docking. In *International Conference on Learning Representations*.

Davies, M., Nowotka, M., Papadatos, G., Dedman, N., Gaulton, A., Atkinson, F., Bellis, L., and Overington, J. P. (2015). ChEMBL web services: streamlining

access to drug discovery data and utilities. *Nucleic Acids Research*.

Du, H., Jiang, D., Zhang, O., Wu, Z., Gao, J., Zhang, X., Wang, X., Deng, Y., Kang, Y., Li, D., et al. (2023). A flexible data-free framework for structure-based de novo drug design with reinforcement learning. *Chemical Science*.

Fletcher, R. (2000). *Newton-Like Methods*. John Wiley & Sons.

Francoeur, P. G., Masuda, T., Sunseri, J., Jia, A., Iovanisci, R. B., Snyder, I., and Koes, D. R. (2020). Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of Chemical Information and Modeling*.

Fu, T., Gao, W., Coley, C., and Sun, J. (2022). Reinforced genetic algorithm for structure-based drug design. *Advances in Neural Information Processing Systems*.

Ganea, O.-E., Huang, X., Bunne, C., Bian, Y., Barzilay, R., Jaakkola, T. S., and Krause, A. (2022). Independent SE(3)-equivariant models for end-to-end rigid protein docking. In *International Conference on Learning Representations*.

Garg, V. K., Jegelka, S., and Jaakkola, T. (2020). Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*.

Guan, J., Qian, W. W., Peng, X., Su, Y., Peng, J., and Ma, J. (2023a). 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *International Conference on Learning Representations*.

Guan, J., Zhou, X., Yang, Y., Bao, Y., Peng, J., Ma, J., Liu, Q., Wang, L., and Gu, Q. (2023b). DecompDiff: Diffusion models with decomposed priors for structure-based drug design. In *International Conference on Machine Learning*.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv*.

Hie, B., Candido, S., Lin, Z., Kabeli, O., Rao, R., Smetanin, N., Sercu, T., and Rives, A. (2022). A high-level programming language for generative protein design. *bioRxiv*.

Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. (2022). Equivariant diffusion for molecule generation in 3D. In *International Conference on Machine Learning*.

Igashov, I., Stärk, H., Vignac, C., Schneuing, A., Satorras, V. G., Frossard, P., Welling, M., Bronstein, M., and Correia, B. (2024). Equivariant 3d-conditional diffusion model for molecular linker design. *Nature Machine Intelligence*.

Immonen, J., Souza, A., and Garg, V. (2023). Going beyond persistent homology using persistent homology. *Advances in Neural Information Processing Systems*.

Ingraham, J. B., Baranov, M., Costello, Z., Barber, K. W., Wang, W., Ismail, A., Frappier, V., Lord, D. M., Ng-Thow-Hing, C., Van Vlack, E. R., et al. (2023). Illuminating protein space with a programmable generative model. *Nature*.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. (2012). Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*.

Jegelka, S. (2022). Theory of graph neural networks: Representation and learning. In *The International Congress of Mathematicians*.

Jin, W., Barzilay, R., and Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*.

Jin, W., Yang, K., Barzilay, R., and Jaakkola, T. (2019). Learning multimodal graph-to-graph translation for molecular optimization. In *International Conference on Learning Representations*.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *International Conference on Machine Learning*.

Joshi, C. K., Bodnar, C., Mathis, S. V., Cohen, T., and Lio, P. (2023). On the expressive power of geometric graph neural networks. In *International Conference on Machine Learning*.

Jumper, J. M., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D. A., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*.

Karczewski, R., Souza, A. H., and Garg, V. (2024). On the generalization of equivariant graph neural

networks. In *International Conference on Machine Learning*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Landrum, G., Tosco, P., Kelley, B., sriniker, gedeck, NadineSchneider, Vianello, R., Ric, Dalke, A., Cole, B., AlexanderSavelyev, Swain, M., Turk, S., N, D., Vaucher, A., Kawashima, E., Wójcikowski, M., Probst, D., guillaume godin, Cosgrove, D., Pahl, A., JP, Berenger, F., strets123, JLVarjo, O'Boyle, N., Fuller, P., Jensen, J. H., Sforna, G., and Gavid, D. (2020). Rdkit.

Le, T., Cremer, J., Noe, F., Clevert, D.-A., and Schütt, K. T. (2024). Navigating the design space of equivariant diffusion-based generative models for de novo 3d molecule generation. In *International Conference on Learning Representations*.

Li, Y., Pei, J., and Lai, L. (2021). Structure-based de novo drug design using 3d deep generative models. *Chemical science*.

Lin, H., Huang, Y., Zhang, O., Liu, Y., Wu, L., Li, S., Chen, Z., and Li, S. Z. (2024). Functional-group-based diffusion for pocket-specific molecule generation and elaboration. *Advances in Neural Information Processing Systems*.

Liu, Z., Li, Y., Han, L., Li, J., Liu, J., Zhao, Z., Nie, W., Liu, Y., and Wang, R. (2015). Pdb-wide collection of binding data: current status of the pdbbind database. *Bioinformatics*.

Luo, S., Guan, J., Ma, J., and Peng, J. (2021a). A 3d generative model for structure-based drug design. In *Advances in Neural Information Processing Systems*.

Luo, Y., Yan, K., and Ji, S. (2021b). GraphDF: A discrete flow model for molecular graph generation. In *International Conference on Machine Learning*.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. (2019). Provably powerful graph networks. In *Advances in Neural Information Processing Systems*.

McNutt, A., Francoeur, P., Aggarwal, R., Masuda, T., Meli, R., Ragoza, M., Sunseri, J., and Koes, D. (2021). Gnina 1.0: molecular docking with deep learning. *Journal of Cheminformatics*.

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M., Mosquera, J., Mutowo, P., Nowotka, M., Gordillo-Marañón, M., Hunter, F., Junco, L., Mugumbate, G., Rodriguez-Lopez, M., Atkinson, F., Bosc, N., Radoux, C., Segura-Cabrera, A., Hersey, A., and Leach, A. (2018). ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI Conference on Artificial Intelligence*.

Pandey, M., Fernandez, M., Gentile, F., Isayev, O., Tropsha, A., Stern, A. C., and Cherkasov, A. (2022). The transformational role of GPU computing and deep learning in drug discovery. *Nature Machine Intelligence*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*.

Patterson, D., Cramer, R., Ferguson, A., Clark, R., and Weinberger, L. (1996). Neighborhood behavior: A useful concept for validation of "molecular diversity" descriptors. *Journal of medicinal chemistry*.

Peng, X., Luo, S., Guan, J., Xie, Q., Peng, J., and Ma, J. (2022). Pocket2Mol: Efficient molecular sampling based on 3D protein pockets. In *International Conference on Machine Learning*.

Puny, O., Lim, D., Kiani, B., Maron, H., and Lipman, Y. (2023a). Equivariant polynomials for graph neural networks. In *International Conference on Machine Learning*.

Puny, O., Lim, D., Kiani, B., Maron, H., and Lipman, Y. (2023b). Equivariant polynomials for graph neural networks. In *International Conference on Machine Learning*.

Pushpakom, S., Iorio, F., Eyers, P. A., Escott, K. J., Hopper, S., Wells, A., Doig, A., Guilliams, T., Latimer, J., McNamee, C., et al. (2019). Drug repurposing: progress, challenges and recommendations. *Nature reviews Drug discovery*.

Ragoza, M., Masuda, T., and Koes, D. (2022). Generating 3d molecules conditional on receptor binding sites with deep generative models. *Chemical Science*.

Sato, R. (2020). A survey on the expressive power of graph neural networks. *arXiv*.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E(n) equivariant graph neural networks. In *International Conference on Machine Learning*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*.

Schneuing, A., Harris, C., Du, Y., Didi, K., Jamasb, A., Igashov, I., Du, W., Gomes, C., Blundell, T. L., Lio, P., et al. (2024). Structure-based drug design with equivariant diffusion models. *Nature Computational Science*.

Shi, C., Wang, C., Lu, J., Zhong, B., and Tang, J. (2023). Protein sequence and structure co-design with equivariant translation. In *International Conference on Learning Representations*.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. (2020). Graphaf: a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*.

Spiegel, J. O. and Durrant, J. D. (2020). Autogrow4: an open-source genetic algorithm for de novo drug design and lead optimization. *Journal of cheminformatics*.

Stärk, H., Beaini, D., Corso, G., Tossou, P., Dallago, C., Günnemann, S., and Lió, P. (2022a). 3D infomax improves GNNs for molecular property prediction. In *International Conference on Machine Learning*.

Stärk, H., Ganea, O., Pattanaik, L., Barzilay, D., and Jaakkola, T. (2022b). EquiBind: Geometric deep learning for drug binding structure prediction. In *International Conference on Machine Learning*.

Steinegger, M. and Söding, J. (2017). Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature Biotechnology*.

Tripp, A. and Hernández-Lobato, J. M. (2023). Genetic algorithms are strong baselines for molecule generation. *arXiv*.

Trott, O. and Olson, A. J. (2010). Autodock vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*.

Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., Li, B., Madabhushi, A., Shah, P., Spitzer, M., et al. (2019). Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Verkuil, R., Kabeli, O., Du, Y., Wicky, B. I. M., Milles, L. F., Dauparas, J., Baker, D., Ovchinnikov, S., Sercu, T., and Rives, A. (2022). Language models generalize beyond natural proteins. *bioRxiv*.

Verma, Y., Heinonen, M., and Garg, V. (2023). Abode: Ab initio antibody design using conjoined odes. In *International Conference on Machine Learning*.

Verma, Y., Kaski, S., Heinonen, M., and Garg, V. (2022). Modular flows: Differential molecular generation. In *Advances in Neural Information Processing Systems*.

Vignac, C., Loukas, A., and Frossard, P. (2020). Building powerful and equivariant graph neural networks with structural message-passing. *Advances in Neural Information Processing Systems*.

Wang, X. and Zhang, M. (2022). How powerful are spectral graph neural networks. In *International Conference on Machine Learning*.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., Wicky, B. I. M., Hanikel, N., Pellock, S. J., Courbet, A., Sheffler, W., Wang, J., Venkatesh, P., Sappington, I., Torres, S. V., Lauko, A., Bortoli, V. D., Mathieu, E., Barzilay, R., Jaakkola, T. S., DiMaio, F., Baek, M., and Baker, D. (2022). Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models. *bioRxiv*.

Wu, K. E., Yang, K. K., van den Berg, R., Alamdari, S., Zou, J. Y., Lu, A. X., and Amini, A. P. (2024). Protein structure generation via folding diffusion. *Nature communications*.

Wu, R., Ding, F., Wang, R., Shen, R., Zhang, X., Luo, S., Su, C., Wu, Z., Xie, Q., Berger, B., Ma, J., and Peng, J. (2022). High-resolution de novo structure prediction from primary sequence. *bioRxiv*.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations*.

Zang, C. and Wang, F. (2020). Moflow: An invertible flow model for generating molecular graphs. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zhang, Z. and Liu, Q. (2023). Learning subpocket prototypes for generalizable structure-based drug design. In *International Conference on Machine Learning*.

Zhang, Z., Min, Y., Zheng, S., and Liu, Q. (2022). Molecule generation for target protein binding with structural motifs. In *International Conference on Learning Representations*.

Checklist

1. For all models and algorithms presented, check if you include:

(a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] Section 4

(b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] Table 1

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] We include code to reproduce our results as part of the submission.

2. For any theoretical claim, check if you include:

(a) Statements of the full set of assumptions of all theoretical results. [Yes] Proposition 3.2 and Lemma 3.1

(b) Complete proofs of all theoretical results. [Yes] Appendix K

(c) Clear explanations of any assumptions. [Yes] Section 3

3. For all figures and tables that present empirical results, check if you include:

(a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] The evaluation procedure is described in Section 5 with more details in Appendix H and the submitted code in the supplementary can be used to reproduce our results.

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] Data is described in Section 5 and model details are provide in Appendices D and F

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] Tables 1, 2, 4, 5 and 7

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] Appendix D.1 and F

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes] The datasets CrossDocked2020, ZINC250k and ChEMBL datasets are properly cited. The MoFlow model that we used is also properly cited.

(b) The license information of the assets, if applicable. [Yes] Appendix M

(c) New assets either in the supplemental material or as a URL, if applicable. [Yes] A newly curated dataset described in Section 4 is included in the submitted supplementary material.

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A DECOUPLING THE UNLABELLED MOLECULAR GRAPH FROM THE ATOM TYPES

Here we provide details of experiments we discuss in section 4.1.

A.1 Vina score

Vina uses a scoring function, which approximates the experimentally measured binding affinity with several types of interactions between the protein and ligand atoms (Trott and Olson, 2010):

$$f(G^P, G^M) = \sum_{k=1}^{K} w_k \sum_{i=1}^{N_P} \sum_{j=1}^{N_M} f_k\Big(a_i^P, a_j^M, \mathbf{s}_i^P, \mathbf{s}_j^M\Big), \tag{8}$$

where $\{f_k\}_{k=1}^K$ represent K types of interatomic interactions that depend on atom types a and 3D coordinates $\mathbf{s}$. The weights $\{w_k\}_{k=1}^K$ are pretrained to reflect experimentally measured affinities. We emphasize that *lower values of f are interpreted as stronger binding.* To estimate the binding affinity for a given protein-ligand complex (G^P, G^M), Vina performs a local search for the lowest-score 3D configuration of the molecule (a.k.a. *redocking*). Specifically, gradient optimization is performed over global translations, global rotations, and torsional rotations $\mathcal{T}(G^M)$ of the ligand G^M to optimise its fit to the protein pocket:

$$\text{Vina}(G^P, G^M) = \min\Big\{f(G^P, \hat{G}^M) : \hat{G}^M \in \mathcal{T}(G^M)\Big\}. \tag{9}$$

Vina uses the BFGS optimiser (Fletcher, 2000). Due to the *redocking* procedure (9), the Vina score can be interpreted as "binding potential", i.e. molecule's binding affinity in its optimal pose.

A.2 Influence of unlabelled molecular graphs on Vina scores

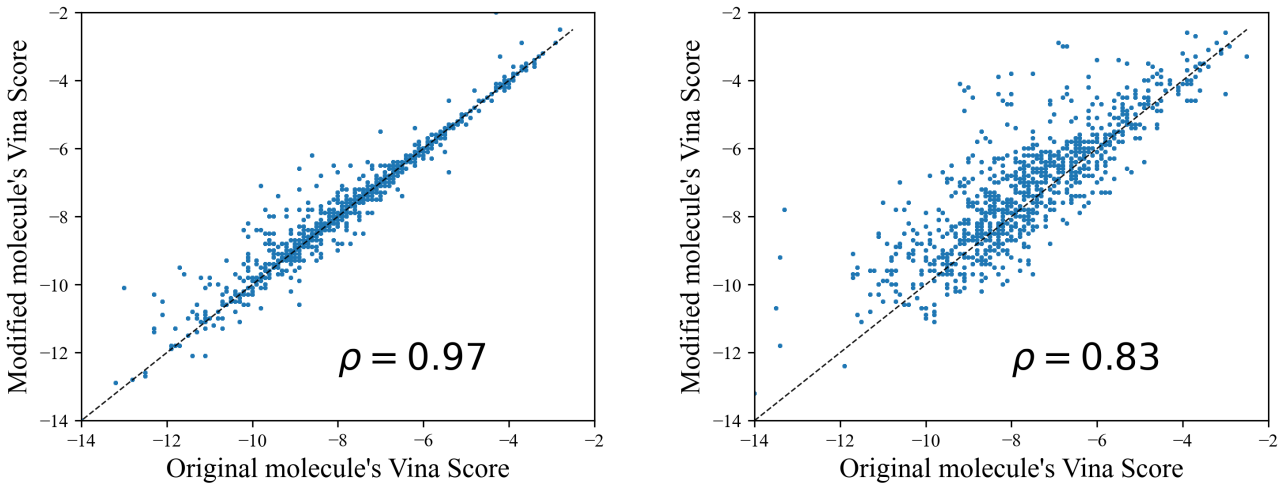
Equation (9) indicates that if Vina employed an ideal optimizer to compute the global minimum, the initial 3D configuration of the molecule would not influence the outcome. Yet, because Vina utilizes an approximate gradient-based optimization method, variations in the initial 3D configuration can indeed affect the Vina score. To measure that, we randomly selected 1,000 protein-ligand complexes from the CrossDocked2020 dataset (Francoeur et al., 2020) and computed the Vina score both before and after modifying the ligand. Here, by "modifying the ligand", we refer to replacing its 3D configuration with a randomly generated conformer using RDKit. As expected, a high correlation at $\rho = 0.97$ emerged indicating an insignificant impact of the ligand's initial 3D configuration on the Vina score (See Figure 5a).

Furthermore, we experimented with changing ligand's atom types. For a fixed unlabelled graph U^M, we re-sample random atom assignments $\mathbf{a}^M$ using MoFlow (Equation 7). This procedure changes the molecule and therefore its 3D configuration is most likely no longer energetically valid. We therefore use RDKit to generate a random conformer of the new molecule and use it to replace the original 3D coordinates. Surprisingly, the correlation remains relatively high at $\rho = 0.83$ (see Figure 5b). These findings suggest that using just the unlabelled graph of the ligand, one can estimate the Vina score with considerable accuracy. We use this observation in the definition of the model.

A.3 Justification of the model decomposition - Gnina

In section 4.1, we justified our model decomposition choice through empirical evidence indicating that the unlabelled molecular graph inherently contains substantial information, as reflected in the Vina score, even when the ligand's initial 3D configuration and atom type assignment are varied. We now show that a similar phenomenon can be observed when, instead of Vina, a reportedly significantly more accurate binding software Gnina (McNutt et al., 2021) is used.

We repeat the experiments from the section A.2 with Gnina instead of Vina and find that for changing the initial 3D configuration, there is even smaller impact indicated by $\rho = 0.99$ (due to Gnina also performing redocking). For changing atom type assignments, $\rho = 0.85$ slightly higher than it was for Vina. Please see Figure 6 for details.



(a) Changing initial 3D configuration.

(b) Changing atom types and initial 3D configuration

Figure 5: Impact of different transformations of the ligand on the Vina score.

A.4 Generality of the findings

We have demonstrated that significant portion of binding information, as measured by binding software, is contained in the unlabelled molecular graph. So far, we performed all experiments on the gold-standard dataset for SBBD - CrossDocked2020. We now repeat the analysis from section 4.1 for PDBind (Liu et al., 2015), specifically its *refined* subset containing ~ 5000 highest-quality protein-ligand complexes. Arguably, this is a better benchmark than CrossDocked2020, which has been augmented with protein-ligand complexes, without experimentally measured binding (the "cross-docking" procedure).

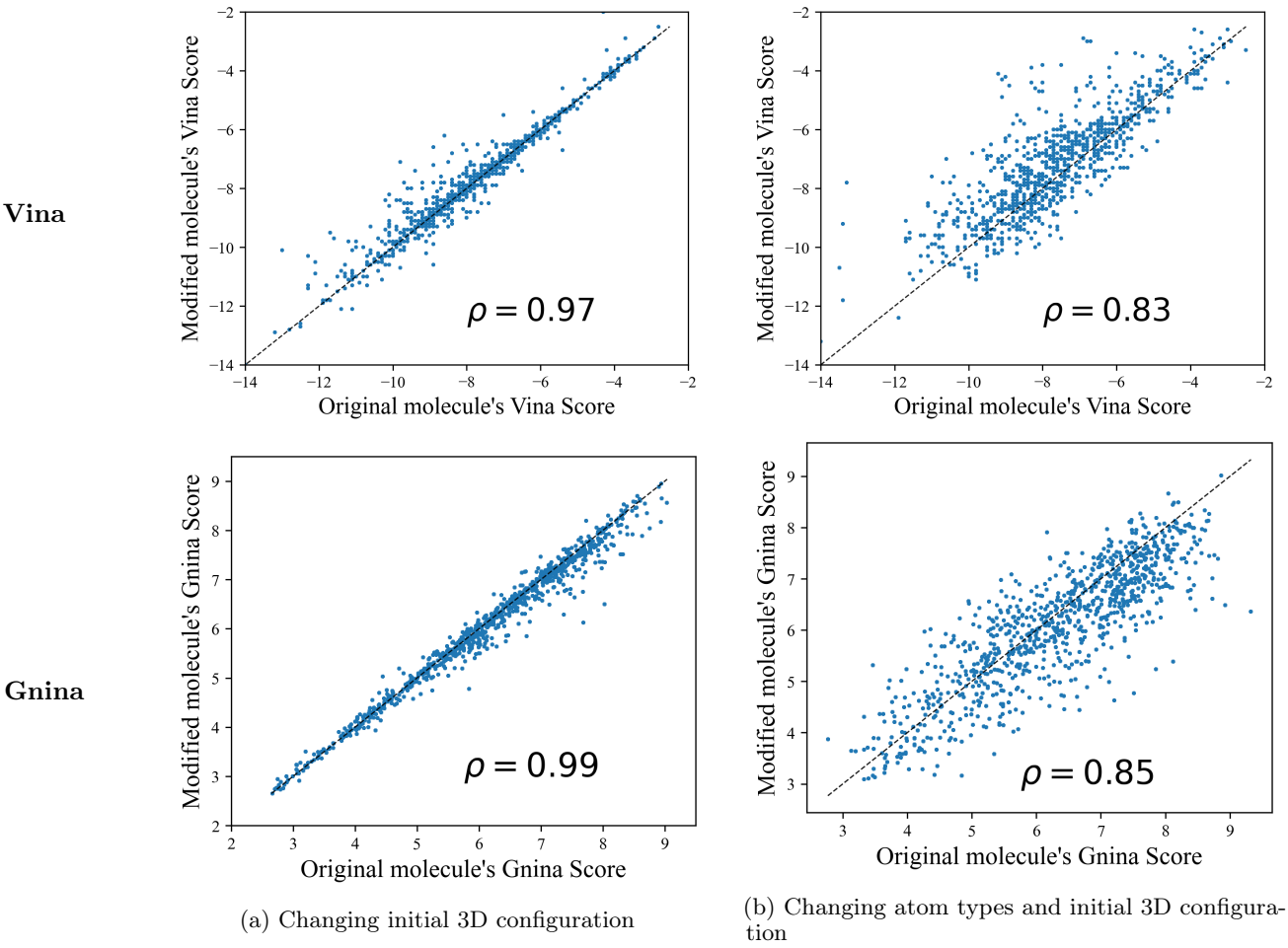
We performed the same experiment as in Section 4.1, i.e. measuring the impact of 1) changing the initial 3D configuration, and 2) changing atom types. In both cases, we measured the binding with both Vina and Gnina. We observed that the correlation is slightly lower, but still large enough to infer that a significant portion of the binding information is contained in the unlabelled graph. This suggests that our findings hold more broadly across different datasets, including those compiled from experimental measurements. Please see Table 3 for more details.

Table 3: Unlabelled molecular graph contains a significant portion of binding information. Findings hold across different binding software and datasets.

Experiment	Software	CrossDocked2020	PDBind
Changing atom types	Vina	0.827	0.781
	Gnina	0.849	0.797
Changing intitial 3D configuration	Vina	0.971	0.953
	Gnina	0.989	0.979

B TRADEOFF BETWEEN BINDING AFFINITY PREDICTION AND QED

As seen in Equation (5), our method allows for controlling the binding affinity of the generated ligands. We therefore investigated the performance of the model when we vary the $v_{\min}, v_{\max}$ thresholds. Specifically, we evaluated 10 different versions of the unlabelled graph sampler with $(v_{\min}, v_{\max})$ set to $(q_0, q_5), (q_5, q_{10}), \ldots, (q_{45}, q_{50})$, where q_k is the k^{th} percentile of the predictions for $\mathcal{U}^M$ (note that these percentiles depend on the protein pocket). We note a trade-off between QED and Binding Affinity of these models, but all of them are still better on both of these criteria than the samples from the CrossDocked2020 dataset. We show this in Figure 7. We set $v_{\min} = q_5$



(a) Changing initial 3D configuration

(b) Changing atom types and initial 3D configuration

Figure 6: The impact of molecule modifications on binding scores is consistent across Vina and Gnina software.

and $v_{\max} = q_{10}$ in our experiments.

C CHEMICAL DIVERSITY WITH FIXED UNLABELLED GRAPHS

In section 4.2, we introduce the unlabelled graph sampler, which uses a set of existing molecular graphs to define possible 2D unlabelled graph structures that can be sampled, i.e. unseen graph structures cannot be sampled. We now argue that this restriction still leaves substantial flexibility in the chemical space.

We performed the following experiment. We randomly sampled 100 molecules from the ZINC250k dataset. For each molecule, we used its template unlabelled graph to generate novel valid molecules by randomly replacing atom types without violating valency constraints. Whenever we generated a duplicate molecule we tried again. We terminated the procedure when we generated 1000 unique molecules or reached the limit of 10 failed trials due to duplicate generation.

For 98 of these 100 molecules we successfully generated 1000 unique novel molecules sharing the identical 2D unlabelled graph including bond types. For the remaining 2, the procedure terminated after generating 665 and 566 unique novel molecules respectively. This shows that in 98% of the cases we are able to create at least 1000 novel valid molecules while keeping the unlabelled graph fixed.

Furthermore, we computed the average Tanimoto similarity between the original molecule and the newly generated ones sharing the unlabelled graph. We found that the average Tanimoto similarity was 0.28 ± 0.08 across the 100 sampled molecules from ZINC250k. We compared this number with the average pairwise Tanimoto similarity of the 100 molecules original molecules, which was 0.24 (within one standard deviation). We also contrast this

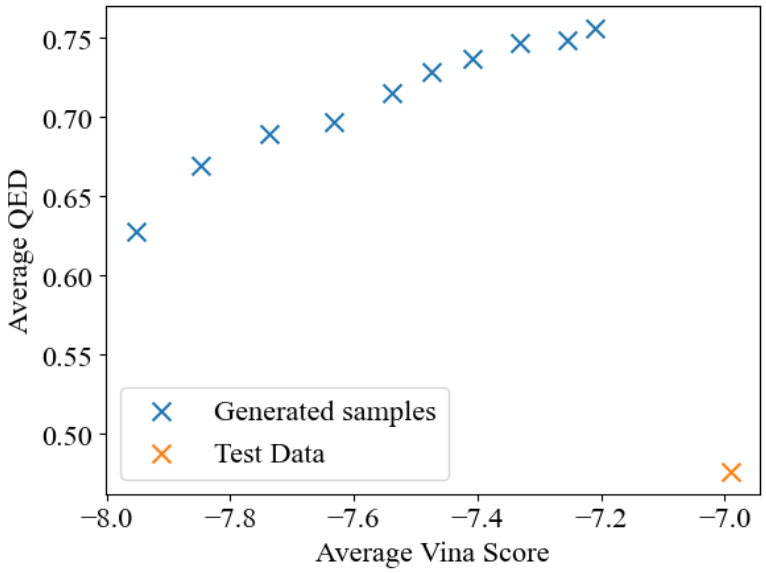


Figure 7: **Binding Affinity vs QED trade-off**. Our model allows for controlling the predicted binding affinity of generated molecules.

number with the 0.85 similarity threshold, above which molecules have a high probability of having the same activity (Patterson et al., 1996).

This shows that, even for a fixed 2D molecular graph with defined bond types, we can define a large number of unique and valid molecules whose similarity to the original molecule is comparable with the pairwise molecular similarity in the ZINC250k dataset and thus proving the substantial chemical variability.

D BINDING AFFINITY APPROXIMATION MODEL DETAILS

In this section we provide more details of the scoring model from Eq. (4). Recall that we want to approximate the binding affinity estimation defined in Eq. (9) using the protein pocket information and ligand's unlabelled graph structure. To that end we represent the protein pocket as a graph with residues as nodes. Each node is equipped with a one-hot encoding of a residue type (1 of 20 possible amino acids) as a feature vector and 3D coordinates computed as a center of mass of the corresponding residue. Two nodes are connected with an edge if they are at most 15Å apart and we limit each node to have at most 24 neighbours. We do not use any edge features. To compute the protein pocket embedding, we use E(3)-Equivariant Graph Neural Networks (EGNNs) (Satorras et al., 2021).

An EGNN layer takes as input a set of node embeddings $\{\mathbf{h}_v^{(l-1)}\}_{v\in V}$, coordinate embeddings $\{\mathbf{x}_v^{(l-1)}\}_{v\in V}$ and edge information $\{e_{uv}\}$ and computes output embeddings $\{\mathbf{h}_v^{(l)}\}_{v\in V}$ $\{\mathbf{x}_v^{(l)}\}_{v\in V}$ for nodes and coordinates respectively. One layer operation can be summarized as follows

$$\begin{aligned}
\mathbf{m}_{u\to v}^{(l-1)} &= \phi_e\left(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}, \|\mathbf{x}_u^{(l-1)} - \mathbf{x}_v^{(l-1)}\|, e_{uv}\right) \\
\mathbf{x}_v^{(l)} &= \mathbf{x}_v^{(l-1)} + C\sum_{u\neq v}\left(\mathbf{x}_v^{(l-1)} - \mathbf{x}_u^{(l-1)}\right)\phi_x(\mathbf{m}_{u\to v}^{(l-1)}) \\
\mathbf{m}_v^{(l-1)} &= \sum_{u\neq v}\mathbf{m}_{u\to v}^{(l-1)} \\
\mathbf{h}_v^{(l)} &= \phi_h(\mathbf{h}_v^{(l-1)}, \mathbf{m}_v^{(l-1)}),
\end{aligned} \tag{10}$$

where ϕ_e, ϕ_x and ϕ_h are learnable functions and $\{\mathbf{h}_v^{(0)}\}_{v\in V}$ are initialized to input node features, i.e. one-hot encodings of amino acid types and $\{\mathbf{x}_v^{(0)}\}_{v\in V}$ are initialized with 3D coordinates of residues' centers of mass. We

encode the protein as the average embedding of the nodes after L layers:

$$\mathbf{h}_P = \frac{1}{|V|} \sum_{v \in V} \mathbf{h}_v^{(L)}, \tag{11}$$

where we use $L = 3$ in our experiments. ϕ_e, ϕ_x and ϕ_h are all implemented as concatenation of the input followed by a fully connected 2-layer MLP with hidden and output dimensions of 16 and the SiLU activation function (Hendrycks and Gimpel, 2016).

We represent the ligand's unlabelled graph structure with the following four features: (i) number of nodes, (ii) number of rings, (iii) number of rotatable bonds and (iv) graph diameter. All these features are standardized by subtracting the mean and dividing by the standard deviation, where these statistics were computed on the train set. The standardized ligand features are subsequently concatenated with $\mathbf{h}_P$ and passed through 5 layers of a fully connected MLP with 50 hidden units, ReLU activation functions and 1 output unit. The model has a total of 15k trainable parameters.

D.1 Training details

The dataset for training the scoring model consists of triples (G^P, G^M, y) representing the protein, ligand and the binding affinity estimated with Vina respectively. We randomly select 10% of the protein pockets for validation and the remaining 90% for training. The model is trained to minimize the mean squared error with the Adam optimizer (Kingma and Ba, 2014) with default parameters of its PyTorch implementation (Paszke et al., 2019). The model was trained with batch size 128 for 10000 steps, after which we did not observe improvements in the validation loss. Model training took approximately 45 minutes on an Apple M1 CPU.

E RDKit POSE GENERATION VARIABILITY

As mentioned in Section 4.3, we use RDKit to generate 3D pose for a molecule given its molecular graph. We now argue that there is substantial variation in the poses generated by RDKit for a fixed molecular graph. Specifically, we performed the following experiment. We randomly sampled 1000 molecules from the ZINC250k dataset. We subsequently generated 20 poses for each and computed their pairwise distances. Where for two poses of the same molecule we define their distance as the lowest possible root-mean-square deviation (RMSD) (computed with `rdkit.AllChem.GetBestRMS(mol1, mol2)`). We used the following pseudocode.

```
mols = random.sample(ZINC250k, k=1000)
variation = []
for mol in mols:
    poses = [generate_pose(mol) for i in range(20)]
    variation.append(average_pairwise_distance(poses))
mean(variation), std(variation)
```

We found that the average pairwise distance between generated poses is 1.60 Å ± 0.46 Å. This average value is higher than e.g. the lengths of C-C, C-H, or C-F bonds suggesting that there is considerable variation in the poses generated by RDKit and it is not merely performing rigid transformations of the same pose.

F LIGAND'S CENTER OF MASS PREDICTOR

As mentioned in Section 4.3, in order to accurately estimate the binding affinity for a protein-ligand pair with Vina, the ligand's 3D configuration must be in the vicinity of the protein 3D configuration. The reason for that is that the scoring function defined in Eq. (8) vanishes for atom pairs that are far apart (Trott and Olson, 2010). Consequently, gradient based optimization used to find the best 3D configuration (Eq. (9)) will return the original configuration, because all gradients are zero whenever the protein and ligand are too far apart.

To that end, we train a model, which predicts the center of mass of ligand's 3D configuration based on the protein pocket. We subsequently use the model to place the ligands' 3D configurations computed with RDKit at the predicted center of mass. As model parametrization we use the EGNN as described in Appendix D. The model

output is defined as:

$$\hat{\mathbf{x}} = \frac{1}{|V|} \sum_{v \in V} \mathbf{x}_v^{(L)}, \tag{12}$$

where $\mathbf{x}_v^{(L)}$ is defined in Eq. (10). As shown in (Satorras et al., 2021), $\hat{\mathbf{x}}$ is then equivariant to translations and rotations as required for the center of mass predictor model. In our experiments, we used $L = 4$ layers and a hidden dimension of 16. The model has 8k trainable parameters and *both its number of parameters and runtime were included when reporting the results for SimpleSBDD models in Table 1.*

To train the model, we used the training protein-ligand pairs of the CrossDocked2020 (Francoeur et al., 2020) dataset described in Section 5. We further split them into train and validation sets to monitor overfitting using a 85-15 train-validation split. Similarly to (Peng et al., 2022), the split was performed to ensure that all validation protein pockets will have sequence similarity lower than 30% to all training protein pockets.

The model was trained with Adam optimizer (Kingma and Ba, 2014) with default parameters of its PyTorch implementation (Paszke et al., 2019) to minimize the Euclidean distance between the predicted and ground truth centers of mass. We used a batch size of 64 and trained the model for 10000 steps until the validation loss converged to ~ 1.15Å. Training took approximately 25 minutes on an Apple M1 CPU.

G ADDITIONAL MOLECULAR METRICS

Here we provide additional metrics to complement the ones reported in Table 1, which were omitted for presentation clarity. Namely, we include the Lipinski's Rule of Five and LogP, which were defined in Section 2. For Lipinski's Rule of Five, 5 is the highest possible value. For LogP we do not indicate which method is the best as there is no straightforward criterion to compare LogP values. In (Peng et al., 2022) it is noted that values in the $(-0.4, 5.6)$ interval are considered good drug candidates. Note that for SimpleSBDD, the value of LogP is the closest to the midpoint of that interval (2.6). None of the other baseline methods reported LogP or the Lipinski's Rule of Five.

G.1 Novelty of generated molecules

In Section 5 we report the novelty of all the methods as the average Tanimoto dissimilarity to the most similar molecule in the training set. Since our method also uses ZINC250k dataset, we also measured the novelty when treating the ZINC250k dataset as the training set. We differentiate between these two metrics as $\text{Novelty}_{\text{CD}}$ (treating CrossDocked2020 as the training set) and $\text{Novelty}_{\text{Z}}$ (treating ZINC250K as the training set). We report the numbers in Table 4.

Table 4: Additional metrics. Error bars correspond to the standard deviation across test protein pockets.

	Lipinski (↑)	LogP	$\text{Novelty}_{\text{CD}}$ (↑)	$\text{Novelty}_{\text{Z}}$ (↑)
Test set	4.34 ± 1.14	0.89 ± 2.73	-	-
DiffSBDD (2024)	4.73 ± 0.69	1.00 ± 1.90	**0.54** ± 0.14	**0.52** ± 0.13
Pocket2Mol (2022)	4.90 ± 0.35	1.71 ± 1.98	0.45 ± 0.16	0.46 ± 0.15
FLAG (2022)	4.94 ± 0.14	0.63 ± 2.38	0.44 ± 0.17	0.46 ± 0.15
DrugGPS (2023)	4.92 ± 0.12	0.91 ± 2.15	0.47 ± 0.15	0.48 ± 0.15
TargetDiff (2023a)	4.59 ± 0.83	1.37 ± 2.37	0.47 ± 0.14	0.46 ± 0.13
DecompDiff (2023b)	4.64 ± 0.90	1.57 ± 2.12	0.52 ± 0.13	0.48 ± 0.12
D3FG (2024)	4.97 ± NA	2.82 ± NA	-	-
EQGAT-diff (2024)	4.66 ± 0.72	-	-	-
SimpleSBDD	4.96 ± 0.20	3.33 ± 1.49	0.51 ± 0.10	0.47 ± 0.10
SimpleSBDD–$\mathcal{PO}$	**5.00** ± 0.03	3.24 ± 0.79	0.50 ± 0.09	0.44 ± 0.09

H BASELINE COMPARISON DETAILS

In this section we provide more details on the baseline comparison reported in Table 1.

Table 5: **Our method allows for simultaneous optimization of binding and other properties**. Comparison of our method w/ and w/o property optimization. With explicit property optimization, we can significantly improve all metrics without sacrificing binding affinity. The increased sampling time is still an order of magnitude lower than baseline methods even though running exclusively on CPU. Error bars correspond to the standard deviation across test protein pockets.

	Vina Score (kcal/mol, ↓)	High Affinity (↑)	QED (↑)	SA (↑)	Diversity (↑)	#Params (↓)	Time (s, ↓)
Test set	-6.99 ± 2.16	-	0.48 ± 0.21	0.73 ± 0.14	-	-	-
SimpleSBDD	-7.78 ± 1.47	0.71 ± 0.34	0.61 ± 0.18	0.69 ± 0.09	0.68 ± 0.06	23K	**3.9**† ± 0.9
SimpleSBDD–$\mathcal{PO}$	**-7.98** ± 1.46	**0.75** ± 0.35	**0.80** ± 0.10	**0.73** ± 0.08	0.67 ± 0.06	23K	115† ± 11

Pocket2Mol We re-evaluated the method using samples generated with a checkpoint available in the official implementation and obtained results very close to the originally reported.

DiffSBDD The authors did not report some of the metrics, so we generated the samples using checkpoints provided by the authors in the original implementation. However, when we tried to use the DiffSBDD-inpaint model variant (reportedly the best one) the sampling produces non-sensical molecules comprising of single atoms. We were able to generate proper samples using the DiffSBDD-cond model variant and this is what we report in all our comparisons.

FLAG & DrugGPS We used the numbers provided in the original publications. The only exception was Novelty (1 - `Sim.Train` in the original publications). We were able to compute Novelty with our code using the samples provided by the authors. The results differ significantly and we suspect that the similarity was computed differently, but we were unable to verify as the code repositories do not contain evaluation code.

TargetDiff & DecompDiff In the manuscripts, the authors use a different version of the Vina software, so we re-evaluate the methods ourselves. For TargetDiff, we use the samples provided by the authors and for DecompDiff, we generate them using the official implementation and provided checkpoints.

EQGAT-diff We copy the numbers from the original publication. We were unable to verify or reproduce the results, because evaluation code is not provided. We were also unable to run the method ourselves, because as the authors say in the official code repository: *Note that currently we do not provide the datasets, hence the code is currently just for reviewing and how the model and training runs are implemented.* Also, the trained model checkpoints are provided on request, but only for unconditional generation models, not condintioned on the protein pocket.

D3FG We copy the results from the original publication, but were unable to verify if the metrics were computed in the same way as there is no implementation available.

I SIMPLESBDD WITH PROPERTY OPTIMIZATION

In this section we provide additional details on the property optimization capabilities of SimpleSBDD described in Section 5.2. Specifically, we show in Table 5 that by the introduction of the additional molecular property procedure, we improve all of the metrics without sacrificing the diversity of the generated molecules. This comes at the additional computational cost, which is still orders of magnitude lower than baselines.

I.1 Ablation study

We analyzed how the number of proposal molecules K impacts the performance. In Section 5.2 we chose $K = 50$ and now investigate what happens for $K = 10, 20, 50$. For this experiment, due to computational constraints, we considered a random subset of 20% of the test protein targets and generated 50 molecules per each (as opposed to the standard of 100). Note that the runtime of property optimization scales linearly with K.

We found that increasing K improves both QED and SA as expected, as the model optimizes for a mixture of these two properties. Furthermore, we found that the choice of K does not seem to affect the predicted

Table 6: Increasing the number of proposals K improves molecular metrics without degrading the predicted binding affinity.

	Vina Score (↓)	High Affinity (↑)	QED (↑)	SA (↑)	Diversity (↑)
Test set	-6.99 ± 2.16	-	0.48 ± 0.21	0.73 ± 0.14	-
SimpleSBDD–$\mathcal{PO}$ ($K = 10$)	-7.935 ± 1.469	0.705 ± 0.356	0.760 ± 0.107	0.715 ± 0.078	0.651 ± 0.054
SimpleSBDD–$\mathcal{PO}$ ($K = 20$)	-7.934 ± 1.465	0.724 ± 0.352	0.781 ± 0.100	0.724 ± 0.079	0.651 ± 0.055
SimpleSBDD–$\mathcal{PO}$ ($K = 50$)	-7.939 ± 1.462	0.715 ± 0.375	0.796 ± 0.094	0.734 ± 0.077	0.652 ± 0.055

Table 7: **Scoring model generalizes across datasets and quickly finds diverse high quality molecules.** The scoring model g_θ used for drug repurposing. We score the molecular databases: ZINC250k and ChEMBL respectively with g_θ. The best scoring ones are compared with the reference molecule. Error bars correspond to the standard deviation across test protein pockets.

	Vina Score (↓)	QED (↑)	SA (↑)	Diversity (↑)	Time (s, ↓)
Test set	-6.99 ± 2.16	0.48 ± 0.21	0.73 ± 0.14	-	-
g_θ (ZINC)	-7.82 ± 1.47	**0.66** ± 0.15	**0.78** ± 0.08	0.68 ± 0.05	**1.8** ± 0.4
g_θ (ChEMBL)	**-8.03** ± 1.56	0.56 ± 0.15	**0.79** ± 0.08	0.67 ± 0.07	2.6 ± 1.0

binding affinity, which strengthens our point that the optimization of atom types for various properties for a fixed unlabeled graph structure does not degrade the predicted binding affinity. Please see Table 6 for more details.

J DRUG REPURPOSING

As mentioned in Section 5.2, our scoring model can be used for drug repurposing. We found that for each protein pocket it takes around 2 seconds to find 100 diverse molecules with much higher predicted binding affinity and other improved molecular metrics. Furthermore, we repeated the experiment with ChEMBL dataset (Mendez et al., 2018; Davies et al., 2015) and found that g_θ generalizes well across molecular databases. See Table 7 for more details

K REPRESENTATION LIMITS OF GNNS FOR SBDD

In this section we prove claims made in Section 3. We begin with the notion of indistinguishability for LU(3D)-GNNs. Recall that a layer of a LU-GNN network can be summarized as follows

$$\begin{aligned} m_{u\to v}^{(l-1)} &= \phi(h_u^{(l-1)}, e_{uv}) \\ \tilde{h}_v^{(l-1)} &= \mathrm{AGG}\{m_{u\to v}^{(l-1)} \mid u \in N(v)\} \\ h_v^{(l)} &= \mathrm{COMBINE}\{h_v^{(l-1)}, \tilde{h}_v^{(l-1)}\}, \end{aligned} \tag{13}$$

where $h_v^{(l)}$ is the feature vector of node v after applying l layers and $h_v^{(0)} = h_v$ are v's input features. A layer of a LU3D-GNN is defined analogously with messages depending on distances to neighbouring nodes:

$$m_{u\to v}^{(l-1)} = \phi(h_u^{(l-1)}, e_{uv}, \|x_u - x_v\|), \tag{14}$$

where x_v are 3D coordinates of v. Before we discuss indistinguishability, we formally define it.

Definition K.1 (Indistinguishability)**.** We say that graphs $G_1 = (V, E_1)$, $G_2 = (W, E_2)$ are indistinguishable by LU(3D)-GNNs if there exist node orderings $(v_1, v_2, \ldots, v_N)$ and $(w_1, w_2, \ldots, w_N)$ of V and W respectively s.t. for any choice of $\phi, \mathrm{AGG}, \mathrm{COMBINE}$ and network depth L:

$$h_{v_i}^{(l)} = h_{w_i}^{(l)} \quad \forall i = 1, \ldots, N, l = 1, \ldots, L$$

We denote indistinguishability by $G_1 \equiv G_2$ for LU-GNNs and by $G_1 \equiv_{3D} G_2$ for LU3D-GNNs.

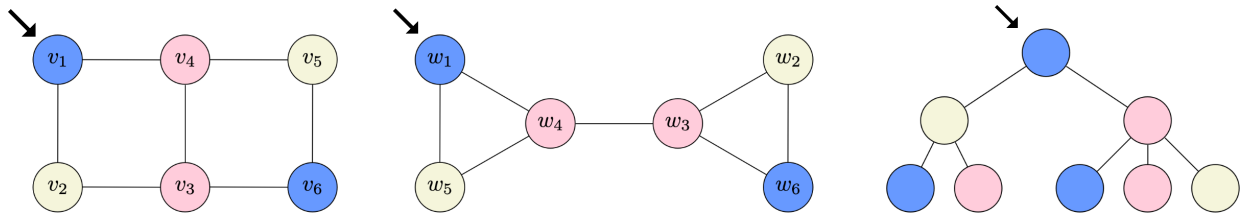


Figure 8: **Construction for Lemma 3.1.** Left and Middle: A pair of non-isomorphic connected graphs differing in all graph properties listed in Lemma 3.1. Right: An example computation tree of depth three, which is identical for nodes v_1 and w_1.

In other words, G_1, G_2 are not distinguishable by LU(3D)-GNNs, if we can pair up their nodes in such a way that corresponding embeddings remain identical in both graphs after any number of layers of any LU(3D)-GNN. We first note that, since LU3D-GNNs are strictly more expressive than LU-GNNs, it holds that:

$$G_1 \equiv_{3D} G_2 \implies G_1 \equiv G_2. \tag{15}$$

A useful construct for determining indistinguishability is that of computation trees (Jegelka, 2022):

Definition K.2 (Computation trees)**.** For a graph $G = (V, E)$, the computation tree of node v is defined recursively as follows:

1. $T_G^{(1)}(v) = h_v$, where h_v is v's feature vector,
2. for $l \geq 2$, the root of $T_G^{(l)}(v)$ is h_v, its children are $\{T_G^{(l-1)}(u) \mid u \in N(v)\}$ and the edge connecting the root to $T_G^{(l-1)}(u)$ inherits features e_{vu} from G.

We denote the multiset of depth-L computation trees of G as $\{T_G^{(L)}(v)\}_{v \in V}$.

Since the multiset of depth-L computation trees represents all information LU-GNNs can extract (Jegelka, 2022), we see that

$$G_1 \equiv G_2 \Leftrightarrow \forall_L \ \{T_{G_1}^{(L)}(v)\}_{v \in V_1} = \{T_{G_2}^{(L)}(v)\}_{v \in V_2} \tag{16}$$

and analogously for $\equiv_{3D}$ whenever the edges of the computation trees carry the information about the distances between nodes. Using the computation tree terminology, we can now prove Lemma 3.1. We recall it for completeness:

Lemma 3.1. *There exist connected non-isomorphic geometric graphs that differ in the number of conjoined cycles, girth, size of the largest cycle and cut-edges that LU3D-GNNs cannot distinguish.*

Proof. In Figure 8 (left and middle), we provide an example of a pair of non-isomorphic connected graphs, which differ in all mentioned graph properties. Colors indicate feature vectors: nodes with the same color have identical features. Edge features can be defined as any function of their endpoints, i.e. $e_{uv} = f(h_u, h_v)$ for any f. All edges are of equal length. For clarity of presentation, the graphs are presented in 2D, but any 3D configuration preserving equal edge lengths could be used instead.

In the figure, we show node orderings such that $T_{G_1}^{(l)}(v_i) = T_{G_2}^{(l)}(w_i)$ for all $i = 1, \ldots, 6$ and $l \geq 1$. As an example, we show $T_{G_1}^{(3)}(v_1)$ in Figure 8 (Right), which is identical to $T_{G_2}^{(3)}(w_1)$. □

We now move on to proving Proposition 3.2. We recall it for completeness:

Proposition 3.2.

(i) If G_1 and G_2 are indistinguishable for LU-GNNs, then for any graph P, the complex graphs $\mathcal{C}(P, G_1)$, $\mathcal{C}(P, G_2)$ are also indistinguishable for LU-GNNs.

(ii) There exist ligand graphs G_1, G_2 differing in graph properties listed in Lemma 3.1, such that for any protein P, the complexes $\mathcal{C}(P, G_1)$, $\mathcal{C}(P, G_2)$ are indistinguishable for models using LU3D-GNNs or LU-GNNs for intra-ligand and intra-protein message passing and LU-GNNs for inter ligand-protein message passing.

Proof. We begin with proving (i). We will abuse notation slightly and for a graph $G = (V, E)$ write $v \in V$ and $v \in G$ interchangeably. We will also write $N_G(v)$ to denote the neighbourhood of v in G whenever we want to emphasize which graph we are considering. Let $C_i = \mathcal{C}(P, G_i)$ for notation brevity and let $(v_1, \ldots, v_N)$ and $(w_1, \ldots, w_N)$ be node orderings of G_1 and G_2 respectively such that $T^{(l)}_{G_1}(v_i) = T^{(l)}_{G_2}(w_i)$, for all $i = 1, \ldots, N$ and $l \geq 1$. We can find such an ordering, because $G_1 \equiv G_2$. Let now $(p_1, \ldots, p_K)$ be any ordering of nodes of P. We will show that for $(u_1, \ldots, u_{N+K}) = (p_1, \ldots, p_K, v_1, \ldots, v_N)$ and $(u'_1, \ldots, u'_{N+K}) = (p_1, \ldots, p_K, w_1, \ldots, w_N)$ the following holds:

$$T^{(l)}_{C_1}(u_i) = T^{(l)}_{C_2}(u'_i) \quad \forall l \geq 1. \tag{17}$$

We start with an auxiliary fact that for all $i, j = 1, \ldots, N$ and any $l \geq 1$ the following holds:

$$\left(T^{(l)}_{G_1}(v_i) = T^{(l)}_{G_1}(v_j)\right) \Rightarrow T^{(l)}_{C_1}(v_i) = T^{(l)}_{C_1}(v_j). \tag{18}$$

We show (18) by induction. The base case of $l = 1$ is trivial since $T^{(1)}_G(v) = h_v$ for any v and G. Assume now that (18) holds for $k = 1, \ldots, l$ and $T^{(l+1)}_{G_1}(v_i) = T^{(l+1)}_{G_1}(v_j)$. We will show that $T^{(l+1)}_{C_1}(v_i) = T^{(l+1)}_{C_1}(v_j)$. We first note that the roots coincide:

$$\text{ROOT}(T^{(l+1)}_{C_1}(v_i)) = h_{v_i} = h_{v_j} = \text{ROOT}(T^{(l+1)}_{C_1}(v_j)) \tag{19}$$

from the base case. Furthermore

$$\begin{aligned} \text{CHILDREN}\left(T^{(l+1)}_{C_1}(v_i)\right) &= \{T^{(l)}_{C_1}(v) \mid v \in N_{C_1}(v_i)\} \\ &= \{T^{(l)}_{C_1}(v) \mid v \in N_{G_1}(v_i)\} \cup \{T^{(l)}_{C_1}(p) \mid p \in P)\} \end{aligned} \tag{20}$$

Now since $T^{(l+1)}_{G_1}(v_i) = T^{(l+1)}_{G_1}(v_j)$, it holds that:

$$\{T^{(l)}_{G_1}(v) \mid v \in N_{G_1}(v_i)\} = \{T^{(l)}_{G_1}(v) \mid v \in N_{G_1}(v_j)\}$$

and therefore by our induction assumption:

$$\{T^{(l)}_{C_1}(v) \mid v \in N_{G_1}(v_i)\} = \{T^{(l)}_{C_1}(v) \mid v \in N_{G_1}(v_j)\}$$

and subsequently, continuing (20):

$$\begin{aligned} \text{CHILDREN}\left(T^{(l+1)}_{C_1}(v_i)\right) &= \{T^{(l)}_{C_1}(v) \mid v \in N_{C_1}(v_i)\} \\ &= \{T^{(l)}_{C_1}(v) \mid v \in N_{G_1}(v_i)\} \cup \{T^{(l)}_{C_1}(p) \mid p \in P)\} \\ &= \{T^{(l)}_{C_1}(v) \mid v \in N_{G_1}(v_j)\} \cup \{T^{(l)}_{C_1}(p) \mid p \in P)\} \\ &= \{T^{(l)}_{C_1}(v) \mid v \in N_{C_1}(v_j)\} \\ &= \text{CHILDREN}\left(T^{(l+1)}_{C_1}(v_j)\right). \end{aligned} \tag{21}$$

Combining (19) and (21) we have shown that $T^{(l+1)}_{C_1}(v_i) = T^{(l+1)}_{C_1}(v_j)$ and therefore proven (18). We now proceed to show (17) by proving a stronger statement, i.e. that for all $l \geq 1$ and all $i, j = 1, \ldots, N, k = 1, \ldots, K$ the following hold:

$$(1) \quad \left(T^{(l)}_{G_1}(v_i) = T^{(l)}_{G_2}(w_j)\right) \Rightarrow T^{(l)}_{C_1}(v_i) = T^{(l)}_{C_2}(w_j) \tag{22}$$

$$(2) \quad T^{(l)}_{C_1}(p_k) = T^{(l)}_{C_2}(p_k) \tag{23}$$

Again, we proceed with a proof by induction and note that the base case of $l = 1$ is trivial, because all computation trees are just features of roots. Assume now that (22) and (23) hold for $k = 1, \ldots, l$. We will show they also hold for $k = l + 1$. Let us first consider any $k = 1, \ldots, K$ and note that

$$\text{ROOT}\left(T^{(l+1)}_{C_1}(p_k)\right) = h_{p_k} = \text{ROOT}\left(T^{(l+1)}_{C_2}(p_k)\right). \tag{24}$$

And for the children:

$$\begin{aligned}\text{CHILDREN}\left(T_{C_1}^{(l+1)}(p_k)\right) &= \{T_{C_1}^{(l)}(v) \mid v \in N_{C_1}(p_k)\} \\ &= \{T_{C_1}^{(l)}(p) \mid p \in N_P(p_k)\} \cup \{T_{C_1}^{(l)}(v) \mid v \in G_1\}.\end{aligned} \tag{25}$$

Since $G_1 \equiv G_2$, it holds that $\{T_{G_1}^{(l)}(v)\}_{v \in G_1} = \{T_{G_2}^{(l)}(w)\}_{w \in G_2}$ and therefore, by the induction assumption (22):

$$\{T_{C_1}^{(l)}(v) \mid v \in G_1\} = \{T_{C_2}^{(l)}(w) \mid w \in G_2\}. \tag{26}$$

Furthermore, by the induction assumption (23), it holds that

$$\{T_{C_1}^{(l)}(p) \mid p \in N_P(p_k)\} = \{T_{C_2}^{(l)}(p) \mid p \in N_P(p_k)\}. \tag{27}$$

Combining (26) and (27) yields:

$$\begin{aligned}\text{CHILDREN}\left(T_{C_1}^{(l+1)}(p_k)\right) &= \{T_{C_1}^{(l)}(v) \mid v \in N_{C_1}(p_k)\} \\ &= \{T_{C_1}^{(l)}(p) \mid p \in N_P(p_k)\} \cup \{T_{C_1}^{(l)}(v) \mid v \in G_1\} \\ &= \{T_{C_2}^{(l)}(p) \mid p \in N_P(p_k)\ \} \cup \{T_{C_2}^{(l)}(w) \mid w \in G_2\} \\ &= \{T_{C_2}^{(l)}(w) \mid w \in N_{C_2}(p_k)\ \} \\ &= \text{CHILDREN}\left(T_{C_2}^{(l+1)}(p_k)\right)\end{aligned} \tag{28}$$

and thus (24) and (28) imply

$$T_{C_1}^{(l+1)}(p_k) = T_{C_2}^{(l+1)}(p_k). \tag{29}$$

We will now show that $T_{C_1}^{(l+1)}(v_i) = T_{C_2}^{(l+1)}(w_i)$ for all i. Consider now any $i = 1, \ldots, N$ and note that:

$$\text{ROOT}\left(T_{C_1}^{(l+1)}(v_i)\right) = h_{v_i} = h_{w_i} = \text{ROOT}\left(T_{C_2}^{(l+1)}(w_i)\right). \tag{30}$$

Moreover:

$$\begin{aligned}\text{CHILDREN}\left(T_{C_1}^{(l+1)}(v_i)\right) &= \{T_{C_1}^{(l)}(v) \mid v \in N_{C_1}(v_i)\} \\ &= \{T_{C_1}^{(l)}(v) \mid v \in N_{G_1}(v_i)\} \cup \{T_{C_1}^{(l)}(p) \mid p \in P\}.\end{aligned} \tag{31}$$

Since $G_1 \equiv G_2$, it holds that $T_{G_1}^{(l+1)}(v_i) = T_{G_2}^{(l+1)}(w_i)$ and therefore
$\{T_{G_1}^{(l)}(v) \mid v \in N_{G_1}(v_i)\} = \{T_{G_2}^{(l)}(w) \mid w \in N_{G_2}(w_i)\}$, so using the induction assumption (22):

$$\{T_{C_1}^{(l)}(v) \mid v \in N_{G_1}(v_i)\} = \{T_{C_2}^{(l)}(w) \mid w \in N_{G_2}(w_i)\}. \tag{32}$$

On the other hand, from the induction assumption (23):

$$\{T_{C_1}^{(l)}(p) \mid p \in P\} = \{T_{C_2}^{(l)}(p) \mid p \in P\}, \tag{33}$$

so from (32) and (33), we have:

$$\begin{aligned}\text{CHILDREN}\left(T_{C_1}^{(l+1)}(v_i)\right) &= \{T_{C_1}^{(l)}(v) \mid v \in N_{C_1}(v_i)\} \\ &= \{T_{C_1}^{(l)}(v) \mid v \in N_{G_1}(v_i)\} \cup \{T_{C_1}^{(l)}(p) \mid p \in P\} \\ &= \{T_{C_2}^{(l)}(w) \mid w \in N_{G_2}(w_i)\} \cup \{T_{C_2}^{(l)}(p) \mid p \in P\} \\ &= \{T_{C_2}^{(l)}(w) \mid v \in N_{C_2}(w_i)\} \\ &= \text{CHILDREN}\left(T_{C_2}^{(l+1)}(w_i)\right).\end{aligned} \tag{34}$$

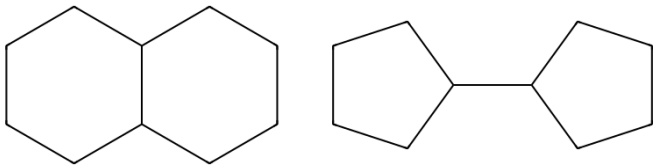


Figure 9: Molecular graphs indistinguishable by LU-GNNs.

Using (30) and (34), we have shown that for all $i = 1 \ldots, N$

$$T_{C_1}^{(l+1)}(v_i) = T_{C_2}^{(l+1)}(w_i). \tag{35}$$

Now take any $i, j = 1, \ldots, N$, such that $T_{G_1}^{(l+1)}(v_i) = T_{G_2}^{(l+1)}(w_j)$. Since $G_1 \equiv G_2$, it holds that $T_{G_1}^{(l+1)}(v_j) = T_{G_2}^{(l+1)}(w_j) = T_{G_1}^{(l+1)}(v_i)$ and thus:

$$T_{C_1}^{(l+1)}(v_i) \overset{(18)}{=} T_{C_1}^{(l+1)}(v_j) \overset{(35)}{=} T_{C_2}^{(l+1)}(w_j). \tag{36}$$

Putting together (36) and (29) concludes the induction step and therefore the proof of (22) and (23). An immediate consequence of (22) is that for all $i = 1, \ldots, N$ and $l \geq 1$:

$$T_{C_1}^{(l)}(v_i) = T_{C_2}^{(l)}(w_i), \tag{37}$$

which together with (23) show that for all $i = 1, \ldots, K + N$ and $l \geq 1$:

$$T_{C_1}^{(l)}(u_i) = T_{C_2}^{(l)}(u_i') \tag{38}$$

and thus $C_1 \equiv C_2$, concluding the proof of (i).

To show (ii), we take G_1 and G_2 to be molecular graphs corresponding to SMILES representations: C1CCC2CCCCC2C1 and C1CCC(C1)C2CCCC2 respectively (see Figure 9). All heavy atoms are carbon and all bonds are single, so all node and edge features are identical. All bonds are also of equal length. We also take an arbitrary protein graph P to define complexes $C_1 = \mathcal{C}(P, G_1)$, $C_2 = \mathcal{C}(P, G_2)$.

The intra-ligand LU3D-GNN produces identical sets of embeddings for G_1 and G_2, which can be shown in the same way as in Lemma 3.1. Similarly for intra-ligand LU-GNNs since $G_1 \equiv_{3D} G_2 \Rightarrow G_1 \equiv G_2$.

The induced subgraphs of C_1 and C_2 corresponding to nodes of P are identical and therefore any intra-protein GNN will produce identical sets of embeddings for both.

Now since $G_1 \equiv G_2$, it follows from (i) that $C_1 \equiv C_2$. Therefore inter protein-ligand LU-GNN will produce identical embeddings for C_1 and C_2. □

K.1 Persistent Homology

Our claims in Propostion 3.2 also hold when the LUGNN model is additionally equipped with persistent homology (PH) information, which makes it strictly more expressive (Immonen et al., 2023). This is a strict generalization, because PH computes global topological information such as the number of connected components or the number of independent cycles that LUGNNs cannot compute.

Specifically, we can show that if the two ligand graphs are indistinguishable by LUGNNs+PH then the complex graphs are also indistinguishable with LUGNNs+PH.

Proof Graphs G_1 and G_2 are indistinguishable by LUGNN+PH if and only if: 1) G_1 and G_2 are indistinguishable by LUGNN, and 2) G_1 and G_2 have the same Betti numbers β_0, β_1, where

- $\beta_0(G)$ is the number connected components of G, and
- $\beta_1(G) = \beta_0(G) + |E| - |V|$, where $G = (V, E)$, i.e. V is the set of vertices and E is the set of edges.

We now assume that G_1 and G_2 are indistinguishable by LUGNNs and $\beta_0(G_1) = \beta_0(G_2)$ and $\beta_1(G_1) = \beta_1(G_2)$.

In proposition 3.2 we already showed that the complex graphs C_1 and C_2 are indistinguishable by LUGNNs, so it suffices to show that C_1 and C_2 have the same Betti numbers.

From our definition of the complex graph we see that it is always connected, i.e. has a single connected component, and thus $\beta_0(C_1) = \beta_0(C_2) = 1$.

To evaluate the 1-st order Betti numbers we denote $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, $C_i = \mathcal{C}(P, G_i)$ for some arbitrary protein graph $P = (V_P, E_P)$ using the notation from the paper.

Since G_1 and G_2 are indistinguishable by LUGNNs+PH, we must have $|V_1| = |V_2|$ and $|E_1| = |E_2|$. Therefore

$$\begin{aligned}\beta_1(C_1) &= \beta_0(C_1) + |E_P| + |E_1| + |E_P| \cdot |E_1| - |V_P| - |V_1| \\ &= 1 + |E_P| + |E_2| + |E_P| \cdot |E_2| - |V_P| - |V_2| \\ &= \beta_1(C_2)\end{aligned}$$

and thus C_1 and C_2 are indistinguishable by LUGNNs+PH.

L EMPIRICAL DEMONSTRATION OF THE THEORETICAL ANALYSIS

In Section 3 we provide a theoretical analysis of the expressivity of Graph Neural Networks (GNNs) and we show that there are inherent limitations in what a GNN can represent for 2-body systems like protein-ligand complexes. As part of our contribution, we defined a scoring model, which estimates the docking scores based on the unlabelled graph structure and the target protein (Section 4). Here, we provide an example of a protein pocket and two ligands such that:

- They have different unlabelled graph structures
- They differ in the predicted docking scores to the given protein pocket
- They are identical from the perspective of LUGNNs

The protein has an identifier
`BAZ2A_HUMAN_1795_1898_0/5mgl_A_rec_5mgl_7mu_lig_tt_min_0_pocket10.pdb` and the two ligands are

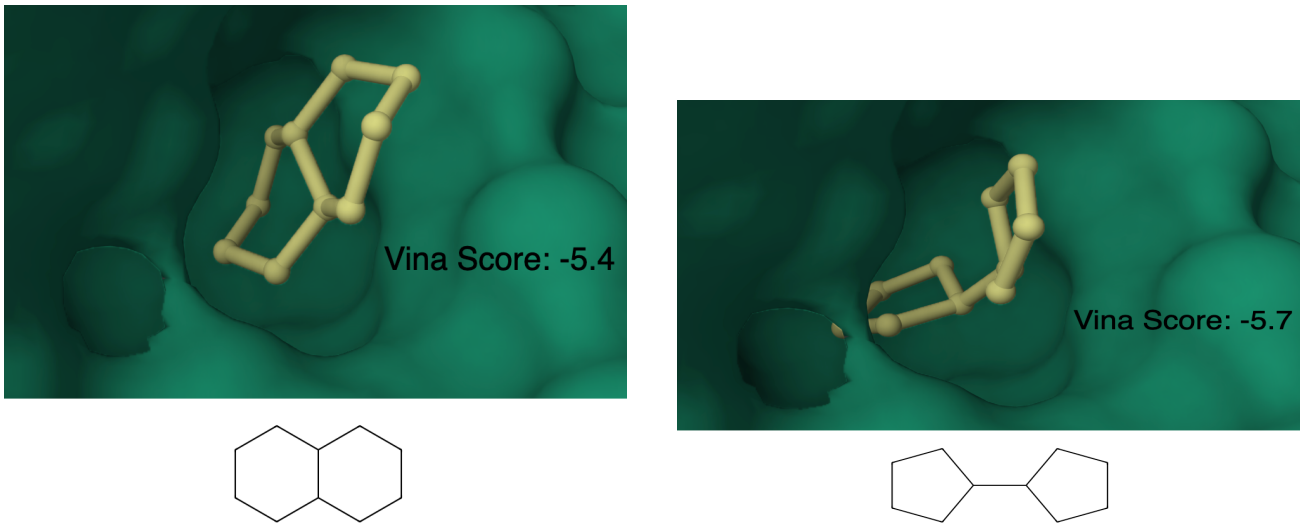
- SMILES: C1CCC2CCCCC2C1 (two 6-rings, zero 5-rings); Vina score -5.4
- SMILES: C1CCC(C1)C2CCCC2 (zero 6-rings, two 5-rings); Vina score -5.7

Even though these two ligands have different binding scores to the given protein, our analysis shows that if we defined the scoring model as a GNN, it would always predict the same value for both of them. See Figure 10

M LICENSES

Datasets

1. CrossDocked2020 (Francoeur et al., 2020): GPL-2.0 license
2. ZINC250k (Irwin et al., 2012): GPL-3.0+ license
3. ChEMBL (Mendez et al., 2018; Davies et al., 2015): CC BY-SA 3.0



(a) SMILES: C1CCC2CCCCC2C1

(b) SMILES: C1CCC(C1)C2CCCC2

Figure 10: An example of two different molecules with different binding affinities, which are identical from LU-GNN perspective.

Pre-trained models

1. MoFlow (Zang and Wang, 2020): MIT License
2. DiffSBDD (Schneuing et al., 2024): MIT License
3. Pocket2Mol (Peng et al., 2022): MIT License
4. TargetDiff (Guan et al., 2023a): MIT License
5. DecompDiff (Guan et al., 2023b): CC BY-NC 4.0
6. RGA (Fu et al., 2022): License status unclear
7. 3D-MCTS (Du et al., 2023): License status unclear
8. Autogrow4 (Spiegel and Durrant, 2020): Apache License Version 2.0

Chemical software

1. RDKit (Landrum et al., 2020): BSD 3-Clause License
2. Vina (Alhossary et al., 2015): Apache License Version 2.0
3. Gnina (McNutt et al., 2021): GPL-2.0 license