TUM Data Innovation Lab

Munich Data Science Institute (MDSI)

Technical University of Munich

&

MIT CS and AI Laboratory

&

TUM Chair of Bioinformatics

Final report of project:

Diffusion Models for Rigid Protein-Protein Docking and Binding Pocket Conditioned Receptor Flexibility

Authors	Simon Dobers, Mohamed Amine Ketata, Cedrik Laue, Ruslan Mammadov, Michael Plainer, Marcella Toth
Mentor(s)	Hannes Stärk, M.Sc.
Co-Mentor	Céline Marquet, M.Sc.
Project Lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

Feb 2023

Abstract

Understanding how proteins interact with other molecules or even proteins is crucial to modern biology, with applications ranging from drug discovery to protein design. The recent machine learning method DiffDock has formulated protein-small molecule docking as a generative problem, with significant performance boosts over both traditional and recent deep learning baselines. In this work, we investigated different ways to extend this work and improve its results. In the first part of the report, we modeled the proteins as semi-flexible to account for structural changes upon docking and to predict the amino acid sidechains' conformations near the ligand's proposed binding site. The experiments presented in this report show promising results. This additional task can aid in predicting better ligand poses alongside structural changes during binding, especially when relying on unbound protein structures. Most prominently, we have shown that the underlying model mainly relies on information from atoms near the binding site, allowing us to gain a significant speedup when discarding atoms too far away without losing accuracy. Still, further research into this direction is needed to make firm claims. In the second part of this report, we investigated a different adaption of the diffusion-based algorithm for rigid protein-protein docking: DiffDock-PP, a generative diffusion model that learns to translate and rotate unbound protein structures to their bound poses. We achieve state-of-the-art performance on DIPS with a median C-RMSD of 4.85, outperforming all considered baselines. Additionally, DiffDock-PP is significantly faster than all search-based methods and generates reliable confidence estimates for its predictions.

Contents

1 Introduction

Proteins realize their myriad biological functions through interactions with other biomolecules, such as other proteins or small molecules. These interacting biomolecules (ligands) can recognize proteins (receptors) and bind to them in a specific manner resulting in intricate molecular complexes. The functionality of these biological complexes is substantially influenced by their final structure. Hence, most of the time when a ligand docks to a receptor, the underlying biological function of the complex changes. It is therefore crucial to understand the protein interaction mechanisms and their effects. With this, we can predict the new complex structure, or more specifically, the position and conformation of the protein and the interaction partner after the binding process. Figure 1 visualizes a ligand that has been docked to a protein.

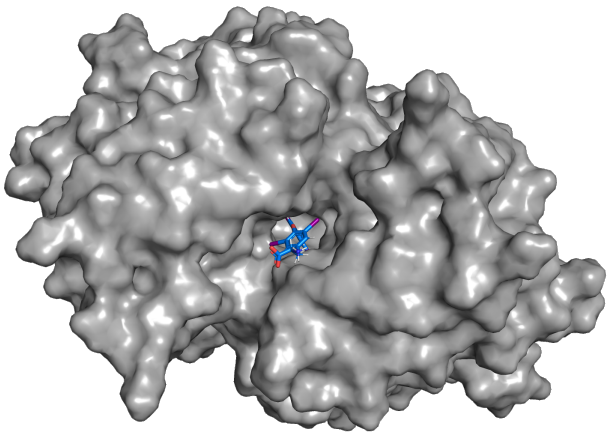


Figure 1: The protein-ligand complex 2rox [61] with a ligand (blue) docked at the binding site (i.e., a cavity) of the protein (gray).

1.1 Background and Project Scope

Most traditional methods for predicting complex structures based on unbound biomolecules [4, 9, 63, 58, 65, 56] rely on optimization algorithms that sample from a large search space and a scoring function that determines the sampled structure's likeness of correctness. Since these sampling algorithms search in a large space and the optimization landscape of the scoring function is usually complicated, generating results is typically a very time-consuming process. In light of recent successes of deep learning, more and more works try to utilize regression-type deep learning models to directly predict protein complex structures [15, 22, 33, 35, 51]. Although these methods are magnitudes faster than classical approaches, they do not achieve similar accuracies.
In contrast to recent regression-based methods, [7] frames the problem as a generative task. As done in [7] for small ligand-protein docking, we argue that for the general task of the molecule (ligand) to protein (receptor) docking, simple one-shot prediction methods only imperfectly serve the objectives of the docking task. Thus given a ligand and a receptor, we try to estimate the distribution over all poses of the ligand using diffusion generative models (DGMs). We then sample from this distribution multiple times and use a trained confidence model to determine the best pose.

The goal of this project was to build upon the work of DiffDock [7] and improve as well as extend it in the following ways:

1. Assess the feasibility of creating a new dataset to improve DiffDock's performance
2. Model protein flexibility during docking
3. Adapt ligand-protein docking to rigid-body protein-protein docking

1.2 Molecular Docking as a Generative Process

Following the arguments of [7] and noting that directly optimizing such thresholding-based objectives is not feasible as they are not differentiable, we argue that these objectives are better aligned with training a *generative* model to maximize the likelihood of the observed structures under the learned distribution. Concretely, since real-world data and complex deep learning models suffer from inherent multi-modal uncertainty, regression-based methods trained to predict a single pose that minimizes an MSE-type loss [15] learn to predict a structure as the weighted mean among many viable alternatives, which is often not biologically plausible. In contrast, a generative model aims to capture the distribution over these alternatives resulting in more accurate and plausible structures.
In generative modeling, the objective is to fit a model to the underlying distribution of a given dataset. This objective allows the synthesis of new data points by sampling from the learned distribution. Representing the probability distribution can be done explicitly or by estimating the score function $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ of the distribution as in [50, 7]. Modeling the score function eludes the necessity of normalization [20] and can come up naturally in regard to diffusion generative modeling [50].
The idea of diffusion generative models is to sequentially add increasing noise to the training data and learn to reverse this process such that new samples of the prior distribution can be generated from random noise. The data perturbation steps can be chosen to form a diffusion process described by the stochastic differential equation (SDE): $dx = f(x,t)dt + g(t)dw$, where $f(\cdot,t)$ is a vector function called the drift coefficient (in [7] always 0), w is a Wiener process (with Gaussian increments of variance dt), and $g(\cdot)$ (a scalar function) the diffusion coefficient of x [50]. Reversing this diffusion process is the key to obtaining new samples. It is important to note that the reverse process corresponds to a closed-form reverse SDE: $dx = [f(x,t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)dw$, which can be derived from the diffusion process SDE and the score function of the modeled probability distribution [50]. Thus it becomes a natural objective to learn the score function as it is sufficient to reverse the diffusion process and generate new samples, as seen in Figure 2.
The performance of a docking system can be measured using the root mean squared deviation (RMSD) between the predicted and ground truth structure. Often only parts of the complex, e.g., ligand or interface, are used to determine this RMSD. Instead of using the raw RMSD value, docking methods are often evaluated based on thresholding [1, 28], for instance, a ligand RMSD < 2 Å might be considered acceptable quality. Score-based diffusion generative models have demonstrated exceptional performance, outperforming previous models in a range of areas, such as image generation [50, 48, 12] audio synthesis [43, 3], music generation [39] or molecular docking [7, 40].

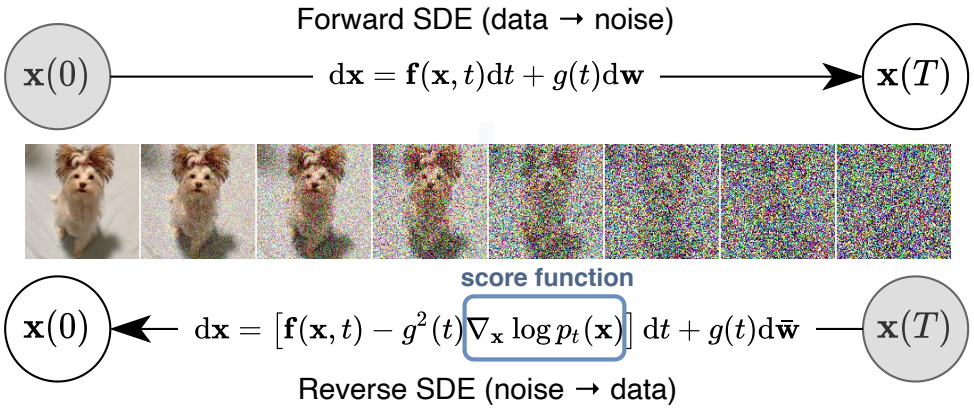


Figure 2: This illustration shows the basis of score-based diffusion. Noise is gradually applied to the data, and the model tries to revert it in the reverse step. This figure was taken from [50, Fig. 1].

Moving forward to molecular docking, for the case of n-atom ligands, the naive approach is to model docking as a diffusion process over $\mathbb{R}^{3n}$. However, a large subset of this space does not represent biologically plausible structures, which suggests modeling the diffusion process only over the main degrees of freedom that are given by rotation, translation, and rotatable bonds of the ligand or more specifically the product space $\mathbb{T}(3) \times SO(3) \times SO(2)^m$ [23, 7]. Intuitively speaking, this approach treats an undocked ligand as a "noisy" version of the binding pose, meaning there exist translation, rotation, and torsional updates that define a transformation between the poses. DiffDock[7] applies this method in a two-step process: 1. Train a *score-model* akin to the above-described methodology that can be used to draw samples from the distribution of docked ligand poses. 2. Train a *confidence-model*, that predicts how physically plausible a given pose is, allowing to selectively choose samples from the score model that are likely to have a low RMSD to the ground truth pose.

With the described methodology, DiffDock has achieved exceptional results in ligand-protein docking. It nearly doubled the previous state-of-the-art deep learning model's performance (from 20% to 38%) and outperforms even the current state-of-the-art search-based methods (23%), while still being three to twelve times faster when run on a GPU. DiffDock also provides an accurate confidence score for its predictions, with 83% of RMSD being less than 2Å in its most confident third of predictions for previously unseen complexes [7]. This project aims to explore *new* use cases of this method and extend its capabilities.

2 Datasets

2.1 Motivation

It is well known that the training data has a significant influence on the model's accuracy. More data can help the model to generalize better, while better quality could ensure that the model can recognize the patterns easier and as a result could improve the model's accuracy. Therefore, we have investigated different existing datasets that could be used for protein-ligand docking to boost the performance of DiffDock.
Our analysis consisted of three steps: Firstly, we searched for existing protein-ligand docking datasets. Then, we analyzed how these datasets were curated, and evaluated their advantages and disadvantages based on the underlying methods. Finally, we analyzed the feasibility of reproducing their curation steps to create our own datasets. The advantage of this approach would be that it would give us more control over the quality of the data and the possibility to generate an up-to-date dataset when needed.

2.2 PDBBind

Most protein-ligand complex datasets (which contain 3D structure information) are derived from the **P**rotein **D**ata **B**ank [2]. **PDB** consists of experimentally derived 3D structures of proteins and protein complexes [2]. Its number of entries is currently over $200,000$, and it is constantly being updated. However, not all of these structures are relevant for protein-ligand docking, therefore filtering methods have been created that use a subset of PDB to create smaller datasets suitable for docking purposes [32, 47, 10]. Some of these methods also process PDB files and curate them with useful information for docking such as binding affinities [47, 10].
One of the most widely used secondary databases for docking is PDBBind, on which DiffDock has also been trained and tested on. It consists of four nested datasets called valid, general, refined, and core set (in 2020 they contained $78,460$, $23,496$, $5,316$, and 285 complexes respectively). The valid complexes dataset filters out complexes from PDB, excluding single proteins and complexes with non-valid ligand heterogens (e.g., inorganic ions, buffer components, or organic solvent molecules). Unfortunately, PDB entries are not properly indicated whether they describe a complex, so the filtering must be done by interpreting the contents of the PDB structure files. In this step, PDBBind selects four major categories of complexes: those formed between protein and small-molecule ligand, between nucleic acid and small-molecule ligand, between two protein molecules, and those formed between protein and nucleic acid. In this first subset of PDB, PDBBind also checks the validity of ligands based on a *hand-curated* dictionary (derived from the Chemical Component Dictionary provided by PDB) [32].
The general set requires further curation by hand, since for every entry, binding affinity information is extracted from the studies listed in the annotation of the PDB files (if there is such information in the cited studies) [32].
A complex set of criteria is considered when selecting the refined set, considering the quality of the complex structures, the quality of the binding data, and the biological and chemical characteristics of the complex. For example, the refined set excludes data obtained by NMR, data with resolution lower than 2.5Å, covalently bonding ligands,

ligands with their mass more than 1000 Da and so on [31].
The core set has been created as a benchmarking dataset for evaluating docking/scoring methods. Its entries are selected in a way that reduces the sample redundancy present in the refined set. For example, nearly 10% of the protein-ligand complexes in the refined set are formed by HIV-1 protease [32].
PDBBind provides a large and diverse collection of curated and validated protein-ligand binding affinity data, making it a valuable resource for computational studies. However, its lack of transparency (e.g., the hand-curated ligand validation dictionary and its selection criteria being unknown) makes its usage less flexible and reliable. The lack of flexibility is also well represented by the fact that the database only gets updated by its maintainers at somewhat arbitrary times. As of writing, the last time PDBBind was updated was 2020 [42], depriving its users of the last three years of complex structure additions of PDB.

2.3 Other Datasets

Additionally, we looked at the Binding MOAD and scPDB datasets. Both datasets are based on PDB but use different methods to extract the ligand-protein complexes from PDB databank [47, 10].

2.3.1 Binding MOAD

In the release of 2020, Binding MOAD dataset contains $41,409$ protein-ligand complexes. For all these complexes, 3D structures are available. Additionally, for $15,223$ of these samples, the binding affinity data is available [47].
Similarly to PDBBind, Binding MOAD is made up of valid complexes extracted from PDB, for which the affinity strength is extracted manually from the literature. However, there are a few differences to PDBBind. Firstly, complexes without affinity strength information are also included in the dataset. Secondly, the extraction algorithm and criteria are different from PDBBind. In particular, only ligands with a resolution better than 2.5Å are considered. Additionally, only complexes with relatively small ligands are included. For example, peptides larger than ten amino acids and chains containing more than four nucleic acids are not considered valid ligands. Therefore, replicating their curation method would give us more control over which complexes get excluded based on the ligand [47].
However, as in the case of PDBBind, the extraction algorithm for Binding MOAD contains both automated and manual curation steps [47]. The main steps are illustrated in Figure 9. As can be seen from this figure, $38,493$ samples needed to be analyzed manually to filter out invalid complexes [47], which may require a lot of human resources and time. Additionally, according to the authors, they have also compiled a list of known crystallographic additives over the course of 20 years. They informed us that they have a list of suspect ligands, which are molecules that can either be additives or genuine ligands (such as citrate) and categorizing these suspect ligands requires a manual evaluation of each structure.

2.3.2 scPDB

In the release of 2017, scPDB dataset contains 16,034 entries. These entries contain 4,782 proteins and 6,326 ligands. As in the case of PDBBind and Binding MOAD, the valid complexes are extracted from PDB. Similarly to Binding MOAD, only small ligands are considered. However, the biggest difference to the previous datasets is that in the case of scPDB, the binding sides are extracted from the complexes. In other words, this dataset's main focus is on the binding mode and not on extracting protein-ligand complexes [10]. The curation steps of the dataset are illustrated in Figure 10. As in the previous datasets, scPDB also contains a manual step—in particular, the ligand is checked for validity [10].

2.4 Issues when Creating Custom Selection Algorithms

Looking through the current existing protein-ligand binding databases, we have come to the conclusion that it would be very beneficial to create an open-source script-based filtering method. Such a project would have many advantages, for example improving data quality by providing a clear overview of the selection criteria and thus also enabling users to select preferred criteria needed for their individual projects. A clear overview of the filtering method would also ensure a consistent and more importantly standardized approach to analyzing protein-ligand complexes. This would make it easier to compare results between studies. When the script does not require manual labor, such a binding dataset could keep up to date with new PDB additions.
However, the reasons which make creating an open-source filtering project highly desirable—such as the current lack of standardization or clear overview of the filtering algorithms and data (including the original PDB structures)—also make this task very difficult. Some descriptions of PDBBind and BindingMOAD methods are vague, and thus their filtering steps cannot be used as instructions for a new method (the authors have not cleared up these uncertain points even upon our inquiries). Because of this, one would have to create a new algorithm that only partially relies on the outlines provided by previous methods which would require in-depth biological understanding and analysis of the filtering steps. Further, the diverse nature of the data produces a lot of edge cases one would have to account for and make well-thought-out decisions on how to handle them. Another issue is that although PDB entries are stored in a standardized format, some inconsistencies can still occur which make the parsing and processing of the data harder [26].
It is crucial to note that the binding datasets we looked at (PDBBind, scPDB and BindingMOAD) rely on hand curation as their last step [47, 32, 10]. This is needed for adding binding affinity, dissociation, and binding coefficient information, which is not necessarily a requirement for our goal. Another reason for it is to overcome inconsistencies in the PDB structure files and even more importantly, to check the validity of the ligands for which even the best standardized computerized methods (used in the previously mentioned datasets) seem to only reduce the need of manual checking but not completely exclude it [47, 32, 10].
Because of the above reasons we decided that the goal of creating a comnpletely script-based open-source filtering algorithm is infeasible during the timeframe of the project and—because of the possibly unavoidable necessity of hand curation required by the large variety and diversity of PDB structure files and biological data—perhaps in general.

3 Binding Pocket Conditioned Receptor Flexibility

In this part of the project, we investigated how a reduced representation of the receptor structure and the explicit modeling of sidechain torsional flexibility influence the success and efficiency of protein-ligand docking.

3.1 Literature and Background

Molecular docking typically uses a rigid protein structure and only explores the conformational space of the ligand. This approach is suitable when the protein structure does not change upon binding. Still, it does not accurately reflect the biochemical processes when the protein conformation (significantly) changes during the docking process and is hence a major limitation [36, 54]. Multiple directions have been studied to overcome this drawback and successfully perform ligand docking. [5] examined 305 distinct proteins with a total of $2,369$ unbound (apo), and $1,679$ bound (holo) structures. They found that conformational changes in the protein backbone upon ligand binding are small compared to changes in the sidechains near the active site. Furthermore, they identified that backbone conformational changes influenced their docking accuracy by less than 0.5Å. The authors of [65] report that 76% of their failed dockings were due to sidechain conformational changes near the active site. By modeling sidechain flexibility, they were able to decrease failed dockings by 60%. Further work about protein-ligand docking also suggests that modeling sidechain flexibility greatly improves the docking success [41, 17]. As modeling flexible sidechains for the full receptor is computationally infeasible [36] and the docking success seems to be primarily influenced by sidechain conformational changes near the binding pocket, modeling flexibility for sidechains within a radius of 3.5Å to the ligand was found to be a reasonable representation for structural changes happening upon ligand binding [38].
To model sidechain flexibility, different methods have been applied. [36, 65, 37, 14, 17] use search-based methods, however, the additional computational cost grows immensely for flexible docking. Other methods use a classical Molecular Dynamics simulation [41], which is unscalable. More recent work also utilizes deep learning methods (GNNs) [35, 33], however, no diffusion models that aim to solve this task are known to us. Modeling those flexible sidechains near the binding site of a ligand (where they are most likely to change), can be useful on its own but more importantly, might increase the success of docking. For the sake of computational feasibility, it makes sense to keep the protein rigid apart from the place where the ligand is likely to dock.

When looking at the 3D structure of a protein, it is often unevenly shaped, or there are "cavities" on the surface (compare Figure 1). Some cavities have the right biochemical conditions that allow a ligand to dock to the so-called *binding pocket* [52]. Having 10–20 different pockets is not uncommon [29], and each ligand interacts differently with each of those pockets. The volume of those binding pockets differs drastically between proteins but is somewhere in the range of 10^2–10^3Å^3 [29].
As only the sidechains near the pockets will be made flexible, prior knowledge of one of these binding pockets is necessary to make those predictions. This position can be analytically determined if a ligand-protein complex has already been analyzed (e.g., is

available in PDBBind). One can simply take the mean of all receptor atoms close enough to the ligand. In the case where the binding pocket is not known ad-hoc, a docking algorithm [7] can be run to compute the most likely ligand pose. In some cases, expert knowledge can also be incorporated to choose pockets of interest. Additionally, each pocket has a radius which is typically the length of the ligand atom farthest away from the pocket.

3.2 Methodology

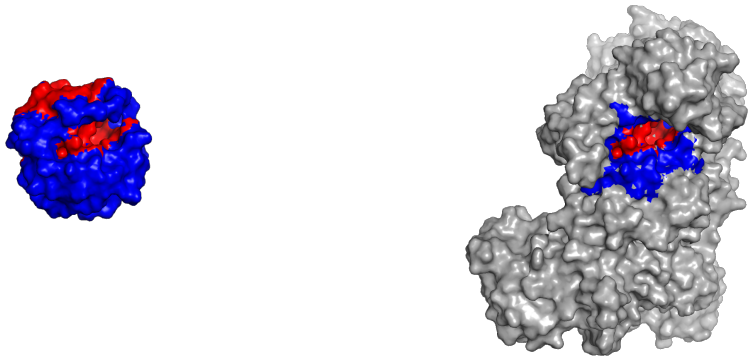
This section will discuss the technical changes made to DiffDock [7], such that it can model sidechain flexibility.

3.2.1 Prior Knowledge of Pocket

As the semi-flexible sidechain prediction requires the specification of a binding pocket, we will discuss different approaches that were implemented to give the model access to this prior knowledge. Namely the *pocket-prior* and the *pocket-reduction* mode. In both cases, the pocket will form the center of the predictions, but the concrete implementations are different. Further, it is important to note that DiffDock relies on a maximal variance that limits the denoising procedure. In all modes where the pocket is known, this maximal σ_{tr} has been reduced so that the model is limited to making predictions near the pocket. The pocket and the two modes are visualized in Figure 3.

Pocket-Prior. This mode was already present in DiffDock and modifies the prior distribution of the diffusion process such that the pocket forms the mean of the distribution. In more constructive words, this means that in [7, Algorithm 2], sampling of $\mathbf{z_{tr}}, \mathbf{z_{tor}}, \mathbf{z_{rot}}$ is done from $\mathcal{N}(p, \Delta\sigma_{tr}^2), \mathcal{N}(p, \Delta\sigma_{rot}^2), \mathcal{N}(p, \Delta\sigma_{tor}^2)$ respectively. p denotes the center of the pocket in this adaption compared to the center of the protein in the initial algorithm.

Pocket-Reduction. This approach is more general and can be applied to a wider variety of different models and was implemented by us. The core idea is that instead of merely shifting the center of the prediction to the center of the pocket, all atoms that a too far from the pocket will be deleted. When predicting the ligand position and pose, this forces the model only to consider atoms that are close to the pocket. As for our model, we assume a rigid protein that is only flexible around the binding pocket, meaning that, in theory, those atoms should not be able to influence them.
DiffDock supports two different modes, the standard, and the all-atom mode. In the standard setup, only the positions of the $C\alpha$ atoms of the protein are used. In this case, the pocket reduction removes all $C\alpha$ atoms (i.e., all residues) beyond a certain threshold distance to the pocket. In the all-atom model, we decided not to discard individual atoms but either keep all atoms of a residue or discard a residue completely. Hence, as long as a single atom of a residue is within proximity of the pocket, it will be retained. This is especially important when modeling flexible sidechains, as this ensures that no rotatable bonds are missing.



(a) Reduced 6agt (b) Full 6agt with pockets

Figure 3: This illustration shows a pocket of the protein 6agt [64]. In Figure 3a, only the atoms near the pocket are visualized, with no additional buffer added to the radius (red) and +8Å added (blue). This part visualizes the pocket-reduction mode, as the remaining part of the protein is removed completely. Figure 3b shows the complete protein on the other hand, illustrating that this protein indeed has a cavity suitable for some molecules to bind to (i.e., a binding pocket). In the pocket-prior mode, the complete protein is available but merely the center is shifted.

3.2.2 Computing Sidechain Torsional Degrees of Freedom

Despite the limited number of possible sidechains (see Figure 12 for a list of possible sidechains), different representations throughout PDB files required us to dynamically determine the rotatable bonds and affiliated subcomponents instead of predefining them. To establish a deterministic and reliable process, we first select each residue that has at least one atom within a specified distance to the pocket center to be made flexible. We compute each selected residue's torsional degrees of freedom by the procedure summarized in algorithm 1. The inputs are all-atom positions of the residue x and their corresponding names $\mathcal{N}$, for instance, [“$C\alpha$“, “$C\gamma$“, ...]. We first filter out non-heavy atoms and construct a directed graph that starts at the $C\alpha$ atom of the sidechain, see Figure 11a for an example. As we ensure the correct edge directions during graph construction, we can then traverse the graph via Breadth-First-Search to find rotatable bonds (see Figure 11b–Figure 11g). A rotatable bond is (in this graph structure) characterized by the graph not being connected after removing the edge and each subcomponent having at least two atoms. For each identified rotatable bond, we store the edge defining this bond in $\mathcal{B}$ alongside with the subgraph containing the second vertex of the bond as a substructure in $\mathcal{M}$. The full subgraph gets rotated around the bond when applying the torsion angles predicted by the model.

3.2.3 Model Adaptation

Once the data preprocessing has been done and the flexible sidechains are determined, the next major step is to adapt the model/loss to predict and learn the positions of the sidechains. The mechanisms and underlying procedure are very similar to the ones used in [7] and originally presented in [23]. Instead of predicting all 3D positions of the atoms, which would be computationally infeasible, we predict the torsions of the rotational bonds

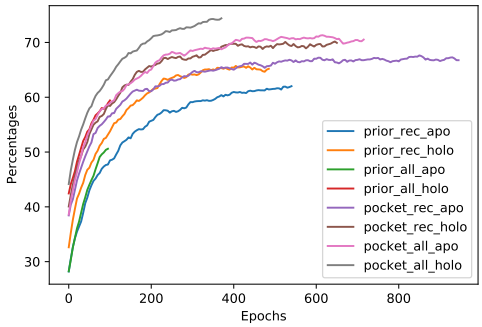
of the sidechains. Those predicted rotations are then applied along the chain of atoms so that the position of the Cα atom stays fixed. The specific implementation is based on $SE(3)$-equivariant convolutional networks [55], with the `e3nn` library [16].
Regarding the specific implementations, since the underlying predictions are similar to the predicted ligand rotations, we relied on a similar architecture. Although, instead of relying on the ligand node attributes, we use the learned embeddings of the atoms. Similarly, we apply a tensor product convolution layer followed by a standard two-layer feed-forward network. The same distances and cutoffs as for the ligand predictions were used. For a complete reference, see the implementation of DiffDock [7].
Although it might seem like the modifications do not influence anything but only predict the sidechain torsions, they will also impact the atoms' representations, which are used with message passing to predict the ligand position. So in principle, modeling the sidechain flexibility can positively impact the expressive power of DiffDock.

3.3 Results

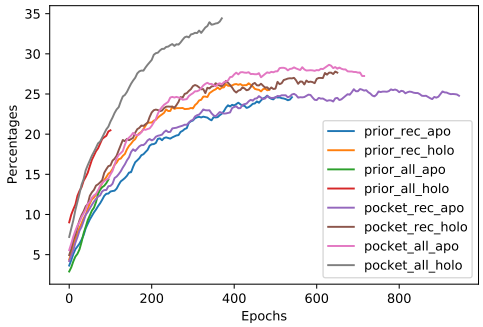
This section is structured so that we first discuss how the different ways to encode pocket knowledge impact performance. Then we will investigate the results of modeling flexible sidechains and discuss the different pocket modes. Due to time and computational constraints, we were not able to evaluate the results on the full models in this section but used a smaller model as a surrogate, see Table 4 for a comparison of model sizes. We have no reason to believe that the presented findings do not generalize to the larger model.
All models have been trained, validated, and tested on the same datasets as DiffDock.

3.3.1 Pocket-Reduced Proteins

Both, changing the prior of the diffusion process to sample from the pocket and removing additional makes learning the ligand pose much easier. This is because the model could learn to predict "0" as a translation, allowing it to have a very low RMSD already. With this in mind, it comes as no surprise that almost all of the predictions have an RMSD of < 5 (compare Figure 4a). What is more important is to analyze the difference between the different pocket modes.
When comparing the empirical results on this small model (see Figure 4), the difference in model performance between the two different pocket modes is noticeable in Figure 4a but diminishes in Figure 4b. This confirms the hypothesis that the model's predictions of the ligand are mostly influenced by atoms close to the binding pocket. Furthermore, one can also see that when relying on the predicted unbound ESMFold [30] structure, the predictions perform significantly worse. This can mainly be credited to the fact that the bound holo-structures from PDBBind already contain a conformation of the protein where the ligand docks; the ESMFold structures, on the other hand, might be very different. Also, while the all-atoms configuration did not make much of a difference for the score model in DiffDock, for all pocket-aware models it did.
In summary, this lets us conclude that the pocket-reduction mode seems superior (at least for the smaller model). As the predictions are on-par or slightly better but take significantly less time. This is also why some training runs were stopped prematurely because the pocket prior runs needed much more computational power.



(a) Percentage of inference RMSDS < 5Å



(b) Percentage of inference RMSDS < 2Å

Figure 4: Both figures show the percentage of ligand pose predictions with an RMSD less than 5Å or 2Å respectively. All combinations of runs with the following parameters (separated by "_") are shown: 1) Whether the pocket reduction or the pocket prior was used 2) whether all atoms or only the Cα atoms of the receptor were used, 3) and if the training was run on the structures from PDBBind (bound structures) or predicted by ESMFold (unbound structures) [30].

3.3.2 Flexible Sidechains

Due to computational constraints, only smaller models (see Table 4) could be trained for 103 epochs. Consequently, we evaluate our implementation of sidechain flexibility in comparison to the rigid *pocket-prior* model (No. 1 in Table 1) instead of the fully trained model from [7].

Table 1: Training results for different combinations of pocket-reduction and sidechain flexibility, each model trained for 103 epochs. SC-RMSD refers to Sidechain-RMSD.

Parameterization					Valinf. RMSD (Å)		Valinf. SC-RMSD (Å)		Epoch Runtime (m)
No.	Pocket-Reduction	Pocket-Cutoff	Flexible Sidechains	Flexdist	%<2	%<5	%<0.5	%<1	Mean
1.	No	-	No	-	16	55	-	-	27.2
2.	Yes	3.5	No	-	18	57	-	-	**10.5**
3.	No	-	Yes	3.5	20	60	29	96	36.8
4.	No	-	Yes	1.5	21	57	**39**	96	36.2
5.	Yes	3.5	Yes	3.5	22	**61**	7	90	12.1
6.	Yes	10	Yes	3.5	**27**	**61**	36	96	20.0

Table 1 shows that all models with sidechain flexibility (No. 3-6) perform better in terms of RMSD for the predicted ligand poses compared to the rigid models (No. 1–2). This lets us conclude that modeling receptor flexibility does not negatively influence the learning of ligand transformations and even suggests a slightly better performance for the docking process. Furthermore, when comparing *pocket-reduced* (No. 2,5,6) to *pocket-prior* models (No. 1,3,4), the average epoch runtimes show that *pocket-reduction* models are, with similar performance, at least 45% faster than *pocket-prior* models.
Another point worth mentioning is that our model with a flexible sidechain radius of 1.5Å has a better sidechain RMSD than the model with a radius of 3.5Å. This is expected, as model No. 4 predicts fewer sidechain torsions, which can be learned after a smaller number of epochs than the larger number of sidechain torsions of model 3. Comparing models 5 and 6 suggests that modeling the full (reduced) receptor is not reasonable, as the

sidechain RMSD <2% is significantly worse for model 5. Instead, reducing the receptor by +10Å and modeling sidechain flexibility for a subset seems to be the most promising approach, as model 6 has the highest percentage of ligand RMSDs below 2% and sidechain RMSDs in the range of model 4 while still realizing a runtime speedup of 45% compared to *pocket-prior* models.

Table 2: Inference Results on the DiffDock testset. Inference was run for 10 samples per complex and numbers are given for (holo/apo) or (bound/unbound) structures. Mean RMSD is given by the average over all predictions and perfect selection RMSD refers to the prediction having the lowest ligand RMSD.

Parameterization					Mean RMSD (Å)				Perfect Selection RMSD (Å)				Runtime (h)
No.	Pocket-Reduction	Pocket-Cutoff	Flexible Sidechains	Flexdist	%<2	%<5	%<1 (sc)	%<2 (sc)	%<2	%<5	%<1 (sc)	%<2 (sc)	
1.	No	-	No	-	6.9 / 3.7	36.3 / 39.9	-	-	30 / 20.5	73.3 / 75.2	-	-	1.2 / 1.3
2.	Yes	3.5	No	-	10.9 / 7.6	57.4 / 53.1	-	-	46.5 / 32.5	93.2 / 89.1	-	-	0.4 / 0.4
3.	No	-	Yes	3.5	1.8 / 1.3	25 / 28.3	0 / -	16.9 / -	10.5 / 9.1	67.2 / 69	0 / -	51.5 / -	3.7 / 3.7
4.	No	-	Yes	1.5	0.8 / 0.7	19.4 / 22.1	0.1 / -	16.6 / -	6.6 / 5.7	57.6 / 63	1.1 / -	49.6 / -	2.3 / 2.3
5.	Yes	3.5	Yes	3.5	6.9 / 4.6	55 / 50.9	0 / -	48.9 / -	44.1 / 23.4	94.1 / 90.3	0 / -	89.6 / -	1 / 0.9
6.	Yes	10	Yes	3.5	5.7 / 3.3	53.2 / 45.7	0 / -	54.9 / -	31.1 / 19.3	93.3 / 87.7	0 / -	93.3 / -	1.3 / 1.3
Gnina	-	-	Yes	3.5	28.3 / 7.4	55.4 / 30.3	100 / -	100 / -	44 / 17.3	82.6 / 64.4	100 / -	100 / -	20 / 165

Table 2 shows the inference results for the models from Table 1 as well as GNINA [36] predictions, are the state-of-the-art baseline for flexible docking. The sc-RMSDs for apo structures are not shown, as currently, no ground-truth sidechains are available. One would have to align each apo sidechain to the holo sidechains akin to the conformer matching procedure of [7] to get ground-truth apo sidechain conformations. Due to the aforementioned model size and training limitations, the shown versions of our models are not fully comparable to the GNINA performances as this would only be reasonable after extensive training and hyperparameter optimization.
Nevertheless, there are already multiple interesting points worth noting. First and foremost, Table 2 shows, that the rigid *pocket-reduction* model (2) outperforms the *pocket-prior* model (1) in terms of prediction quality as well as runtime. When comparing mean and perfect-selection RMSDs, one finds a great discrepancy between the two, suggesting that increasing the number of samples per complex alongside a pocket-reduction confidence model could lead to a great performance increase. Especially the *pocket-reduced* models (flexible and rigid: 2,5,6) already achieve better results on the apo structures for the perfect selection, which leads us to the assumption that these methods could potentially outperform the baseline once being fully optimized. In terms of sidechain RMSD, GNINAs 100%<1 differ from our results, as the sidechains of the holo structures are not randomized before processing, giving GNINA already access to the ground truth structures which are likely to be selected by the scoring function. Besides that, it can be seen, that our *pocket-reduced* flexible models (5,6) achieve a mean sidechain-RMSD < 2 of 50%, which lets us conclude that this might be the most efficient method for flexible docking, as it already shows promising pose and sidechain precisions, comes with a significant runtime speedup and the predicted sidechains can be reinserted to the receptor to retain the full predicted holo structure.
As already pointed out, future work in this direction would include extensive training and hyperparameter optimization for our proposed models in order to assess the possibilities and limitations. Additionally, a specified confidence model should be trained, to improve the choice of predictions. Further, training on the apo structures with conformer matching, as discussed before, also seems like a promising research direction.

4 Protein-Protein

In this part of the report, we focus on the task of rigid protein-protein docking, the undertaking of predicting the structure of protein-protein complexes based on their standalone structure, assuming the rigidity of the proteins' individual backbones during binding. We adapt DiffDock [7] to this task and call this method therefore DiffDock-PP.

4.1 Literature

Protein-Protein Docking. The goal of protein-protein docking is to predict the (bound) structure of a protein-protein complex based on the individual proteins' (unbound) structures. Specifically, we focus on the task of *rigid body* protein-protein docking, which assumes that the proteins do not undergo any deformations during binding, restricting their relative degrees of freedom to a rotation and translation in 3D space. This assumption is often realistic [59] and even leads to improved results for most interacting proteins [11].
To evaluate the quality of the predicted structures, a common approach is to compute the fraction of those lying within some threshold distance to the true bound structure [59, 1, 28].
Search-based Docking Methods. Traditional methods for protein-protein docking usually rely on the physical properties of the complexes [4, 9, 63]. These methods typically 1) generate an initial population of plausible complex structures, 2) further pose proposals using optimization algorithms, and 3) refine the complexes with the highest score according to some scoring function.
Template-based modeling (TBM), which predicts the structure of a target protein by aligning it to one or multiple template proteins with known structures, is also used as a subroutine by some search-based methods [59]. While some of these methods offer decent predictive performance, they are usually computationally expensive and impractical for large-scale molecular screening campaigns.
Deep Learning-based Docking Methods. Deep learning approaches to protein-protein docking can be broadly partitioned into two categories: single-step and multi-step methods. Single-step methods directly predict the complex structure in a one-shot fashion. Notably, [15] proposed EquiDock, a pairwise-independent SE(3)-equivariant graph matching network that directly predicts the relative rigid-body transformation of one of the interacting proteins. In contrast, multi-step methods produce their final predictions by iteratively refining a set of proposed structures. For instance, AlphaFold-multimer [13] was designed to co-fold multiple protein structures, given their primary sequences and multiple sequence alignments (MSAs) to evolutionary-related proteins. Finally, [53] incorporated different physical priors into an energy-based model for predicting protein complex 3D structure.
Our method, DiffDock-PP , naturally falls into the category of multi-step methods due to the multiple steps required to sample from the distribution induced by diffusion generative models. Compared to search-based methods, however, we sample orders of magnitude fewer poses during our refinement process.

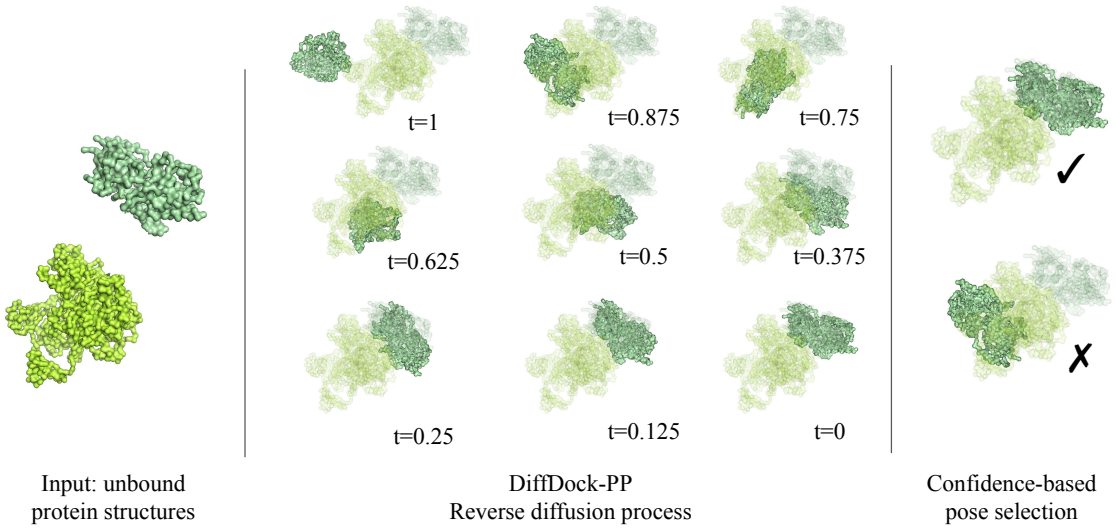


Figure 5: Overview of DiffDock-PP . The model takes two proteins as input, where the ligand has been randomly rotated and translated in 3D space. Then it runs a reverse diffusion process to sample multiple poses. The confidence model ranks these poses, and we output the pose with the highest confidence score. Depicted structure is PDB 1KEE over 40 steps of reverse diffusion.

4.2 Method

4.2.1 Benefits of Generative Modeling for Rigid Protein Docking

Protein-protein docking is often evaluated on the basis of thresholding [1, 28], e.g., a Ligand-RMSD < 5 Å and an Interface-RMSD < 2 Å are among several conditions to consider a given prediction to be of medium quality [28].
Following the arguments of [7] and noting that directly optimizing such thresholding-based objectives is not feasible because they are not differentiable, we argue that these objectives are better aligned with training a *generative* model to maximize the likelihood of the observed structures than with fitting a *regression* model as done in previous work. Concretely, since real-world data, as well as complex deep learning models, suffer from inherent multi-modal uncertainty, regression-based methods trained to predict a single pose that, in expectation, minimizes some MSE-type loss [15] would learn to predict a structure as the weighted mean among many viable alternatives, often not a plausible structure itself. In contrast, a generative model would aim to capture the distribution over these alternatives resulting in more plausible, accurate, and diverse structures.
To illustrate this phenomenon, we visualize some of the structures predicted by our model and compare them to those generated by the baselines, especially EquiDock, which was trained using an MSE-type loss and whose one of the main limitations is the existence of steric clashes in the model's predicted structures [15]. Figure 6 illustrates such predictions. We observe that our model predicts structures with no steric clashes, which we hypothesize is partly due to the adopted generative approach to protein-protein docking.

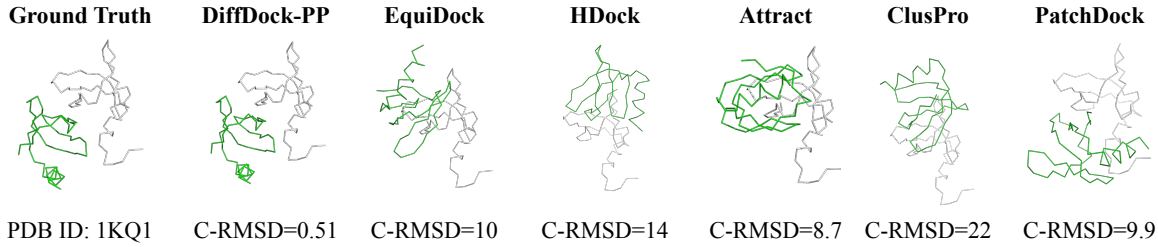


Figure 6: Visualization of the different methods' predictions for a protein complex from our test set.

4.2.2 Method Overview

In rigid protein-protein docking, we aim to predict the complex structure of an interacting protein pair based on the individual structure of each protein.
In this work, we model the proteins on the residue level, representing each protein as a set of amino acid nodes. Each residue is, in turn, represented by its type and the position of its α-carbon atom. We denote $\mathbf{X}_1 \in \mathbb{R}^{3n}$ as the ligand consisting of n residues and the receptor as $\mathbf{X}_2 \in \mathbb{R}^{3m}$ with m residues. The ligand/receptor assignment can, in principle, be arbitrary; however, we define the ligand $\mathbf{X}_1$ as the protein with fewer residues. This means that with $\mathbf{X}_1^* \in \mathbb{R}^{3n}$ and $\mathbf{X}_2^* \in \mathbb{R}^{3m}$ denoting the ground truth complex, the receptor is kept fixed $\mathbf{X}_2 = \mathbf{X}_2^*$ and the task is to predict the structure of the ligand with respect to the receptor.
It is important to note that - since we are considering *rigid-body* protein docking - we only need to consider poses that can be obtained by a *rigid-body* transformation (i.e., a rotation and a translation) of the initial pose $\mathbf{X}_1$. These poses lie on a 6-dimensional submanifold $\mathcal{M} \subset \mathbb{R}^{3n}$ corresponding to the 6 degrees of freedom introduced by the rotation and translation in 3D space. We consider rigid-body protein-protein docking as the task of learning a probability distribution $p(\mathbf{X}_1|\mathbf{X}_2)$ of ligand poses in $\mathcal{M}$, conditioned on the receptor structure $\mathbf{X}_2$.
Now we discuss how we can effectively deploy DGMs to learn this probability distribution. In order to avoid the inefficiencies arising from learning DGMs on arbitrary submanifolds [8], we map the submanifold $\mathcal{M}$ to the manifold defined by the product space of the rotation and translation group analogously to what was done in [7] and define our DGM over this manifold.

4.2.3 Diffusion Process

To formalize the discussion in the previous section, let us introduce the 3D translation group $\mathbb{T}(3)$ and the 3D rotation group $SO(3)$ as well as their product space $\mathbb{P} = \mathbb{T}(3) \times SO(3)$. With this, we can define a rigid-body transformation as the mapping $A : \mathbb{P} \times \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ with

$$A((\boldsymbol{r}, R), \boldsymbol{x})_i = R\,(\boldsymbol{x}_i - \overline{\boldsymbol{x}}) + \overline{\boldsymbol{x}} + \boldsymbol{r}, \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^3$ corresponds to the position of the i-th residue and $\overline{\boldsymbol{x}}$ represents the center of mass of the ligand protein. This equation simply describes a rotation around the center of mass followed by a translation.

The submanifold of ligand poses introduced informally in the previous section can now be described using this transformation as $\mathcal{M} = \{A((\boldsymbol{r}, R), \mathbf{X}_1) \mid (\boldsymbol{r}, R) \in \mathbb{P}\}$. For similar arguments to [7], the map $A(\cdot, \mathbf{X}_1) : \mathbb{P} \to \mathcal{M}$ is a bijection, which guarantees the existence of the inverse map. We can therefore develop a diffusion process over the product space $\mathbb{P}$ to generate a distribution on the manifold.
Given that $\mathbb{P}$ is a product manifold, we can define a forward diffusion process independently on each manifold [44] with the score as an element of the corresponding tangent space [8]. A score model can then be trained with denoising score matching [49]. In both groups, we define the forward SDE as $d\mathbf{x} = \sqrt{d\sigma^2(t)/dt}\, d\mathbf{w}$, where σ^2 is σ^2_{tr} for $\mathbb{T}(3)$ and σ^2_{rot} for $SO(3)$ and $d\mathbf{w}$ denotes the corresponding Brownian motion. The reader is referred to [7] for a description of how we can sample from and compute the score of the diffusion kernel on each of these groups.

4.2.4 Model Architecture

From the discussion in the previous subsection, we see that we need to design the score model $\mathbf{s}(\mathbf{X}_1, \mathbf{X}_2, t)$ such that it produces two outputs that lie in the tangent spaces of the translation and rotation groups. These spaces correspond respectively to translation and rotation (Euler) vectors, which are 3-dimensional $SE(3)$-equivariant vectors. On the other hand, the output of the confidence model $\mathbf{d}(\mathbf{X}_1, \mathbf{X}_2)$ is a single scalar that is $SE(3)$-invariant.
We adapt DiffDock's architecture [7] to mainly account for: i.) the *symmetry* introduced by operating on protein-protein pairs instead of small ligand-protein pairs; we achieve this by using the same input representation for both proteins and using the same set of weights to process them in the early layers of the network, and ii.) the *rigidity* assumption of the proteins, in contrast to the flexibility of the ligand typically assumed in small ligand-protein docking; we achieve this by simply discarding the parts of DiffDock that are responsible for the torsion angles diffusion.
Input representation. Protein structures are represented as heterogeneous geometric graphs with the amino acid residues as nodes. Node features comprise the residue's type, its α-carbon atom's position, as well as language model embeddings trained on protein sequences from ESM2 [30]. In order to construct the edges, we connect each node to its 20 nearest neighbors from the same protein (intra-edges), and we use a dynamic cutoff distance of $(40 + 3 \cdot \sigma_{\text{tr}})$ Å, where σ_{tr} is the current standard deviation of the diffusion translational noise (zero for the confidence model) to connect the nodes from different proteins (cross-edges). The intuition behind using a dynamic cutoff distance is to increase the chances that each node interacts with potentially relevant nodes from the other protein even when the proteins are still far apart (at early diffusion steps) while having a lower computational cost than using a fixed higher cutoff distance (especially at later diffusion steps).
Intermediate layers. Both the score model and the confidence model have similar architectures based on the increasingly popular $SE(3)$-equivariant convolutional networks over point clouds [55, 16]. We follow very closely the architectural details in [7]. Specifically, the model first applies a set of embedding layers to embed and process the initial features, the diffusion time, and the edge lengths. Then, we define a different set of convolutional layers for each edge type (intra-edges and cross-edges). However, in contrast to [7], we

use the same intra-edge layers for both proteins.
Output layers. This is where the main difference between the score and confidence model lies. On the one hand, the score model applies a tensor-product convolution placed at the center of mass of the ligand to produce two 3-dimensional vectors as the translational and rotational scores. On the other hand, the confidence model applies a fully connected layer on the mean-pooled scalar representations from the last convolution layer to produce the confidence value.

4.2.5 Training and Inference

The training and inference regimes follow very closely [7]. We reiterate the most important points here.
Diffusion model. Even though the diffusion kernel and score matching objectives were defined on the product space $\mathbb{P}$, we follow [7] and develop the training and inference procedures directly on ligand poses in 3D space. This should lead to better generalizations and allows the model to reason about physical interactions more easily. Another interesting point to note is that each training example $(\mathbf{X}_1^*, \mathbf{X}_2)$ is the only available sample from the conditional distribution $p(\mathbf{X}_1|\mathbf{X}_2)$. This is unlike the standard generative modeling setting, where many samples are drawn from the same data distribution. Therefore, during training, we iterate over distinct conditional distributions with only one sample. Apart from some details related to torsional diffusion, we otherwise adopt the same training and inference procedure as [7]. During inference, in order to avoid the problem of overdispersed distributions typically observed with generative models, we use low-temperature sampling [21], which allows the model to concentrate on modes with high likelihood.
Confidence model. The training data of the confidence model is generated by sampling the diffusion model for every training example. Based on the generated samples, the corresponding label is then generated via thresholding, i.e., equal to 1 if the RMSD is below a certain threshold and 0 otherwise. In our experiments, we choose the threshold to be 2Å for complex RMSD. We train the binary classification model based on the generated labels.
Combined Inference. We sample multiple candidate poses using reverse diffusion on the score model during inference. These samples are then ranked by the confidence model and sorted in descending order.

4.3 Experimental Setup

4.3.1 Datasets

We evaluate our model in two settings: Database of Interacting Protein Structures (DIPS) [57]. DIPS consists of 42,826 binary protein complexes (no distinction between obligate vs. transient protein interactions). We use the same dataset splits proposed by [15] (protein-family split for DIPS).

4.3.2 Baselines

We compare our method to deep-learning model EquiDock [15], and search-based docking algorithms ClusPro (Piper) [11, 27], Attract [45, 60], PatchDock [34, 46], and

HDock [63, 62, 18, 19]. Due to substantial overlap of AlphaFold-multimer's training corpus with our test sets and the unavailability of code for [53], we primarily consider EquiDock as our deep learning baseline.
To ensure a fair comparison, we follow the evaluation scheme proposed by [15]. All models were evaluated using complex root mean square deviation (CRMSD) and interface root mean square deviation (IRMSD). CRMSD is determined by superimposing the ground truth and predicted complex structures via the Kabsch algorithm [24] and computing the RMSD between all C-α coordinates. IRMSD is determined by similarly aligning both complexes and computing the RMSD over interface C-α coordinates (within 8Å of the binding partner).

4.3.3 Implementation Details

DiffDock-PP was trained on the DIPS train set with the Adam optimizer [25] for a maximum of 170 epochs, with a learning rate of 0.001.
Every 10 epochs, we run reverse diffusion on the DIPS validation set to compute ligand RMSD (LRMSD) and update the best validation model if it achieves a better LRMSD mean and median than the previous one. The best model obtained with this procedure is finally tested on the DIPS test set. During training and inference, the smaller protein is selected as a ligand, and we randomly rotate and translate the ligand in space before running our model.
For both DIPS, we note that except for EquiDock, which used the same data splits as we did, we cannot control the training and testing data of the other baselines. This implies that some of these baselines might have used a part of our test set to train or validate their models. Thus, the reported performance of these methods might be overestimating the true performance.

4.4 Results

Table 3 reports the performances of the different methods on DIPS test set. DiffDock-PP achieves a CRMSD median of 4.85, outperforming all baselines while being more than 4 times faster than the best-performing baseline HDock. When limited to generating one sample for each complex, DiffDock-PP outperforms the majority of the baselines while having the fastest runtime. Specifically, in this setting, it outperforms the deep learning-based approach EquiDock by a large margin while being slightly faster and is very close to HDock's performance while being significantly faster. Ensuring computational efficiency without a significant drop in performance is critical for computational screening applications like drug discovery and antibody design, where one needs to analyze a very large number of complexes.
We note that when we evaluate the model in such a way that we pick the complex with the smallest RMSD from the ones generated by the model, the performance exceeds that of all baselines by a large margin. As such, the performance of DiffDock-PP can be significantly boosted by improving the used confidence model or designing more effective ranking methods.

Table 3: Complex prediction results on 100 samples from the DIPS test set. The last three rows show our method's performance. The number of poses sampled from the generative model is in parenthesis. DIFFDOCK-PP is our model with 1.62M parameters. The methods highlighted with * do not use the same training data as our models and might be using parts of our test sets (e.g., to extract templates or features).

	DIPS Test Set								
	Complex RMSD (Å)				Interface RMSD (Å)				Runtime (s)
Methods	%<2	%<5	%<10	Median	%<2	%<5	%<10	Median	Mean
ATTRACT*	20	23	33	17.17	20	22	38	12.41	1285
HDOCK*	**50**	**50**	50	6.23	**50**	50	58	**3.90**	778
CLUSPRO*	12	27	35	15.77	21	27	42	12.54	10475
PATCHDOCK*	31	32	36	15.25	32	32	42	11.45	7378
EQUIDOCK	0	8	29	13.30	0	12	47	10.19	5
DIFFDOCK-PP (1)	34.2	41	45.6	11.95	35.8	41.6	52.8	8.60	**4.76**
DIFFDOCK-PP (40)	42	**50**	**55**	**4.85**	45	**52**	**63**	4.23	179
DIFFDOCK-PP (40) - oracle	71	79	86	0.67	72	82	91	0.54	179

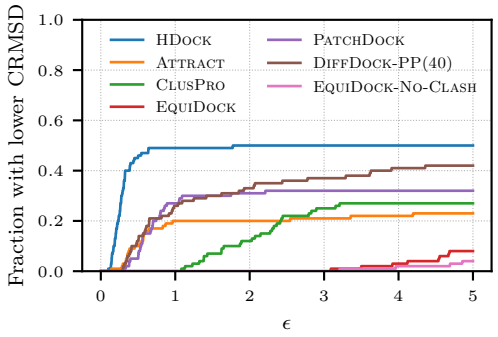


Figure 7: **Fraction of complexes with a CRMSD $< \epsilon$.** Each method's fraction of complexes with an error below a certain CRMSD value ϵ on the DIPS test set.

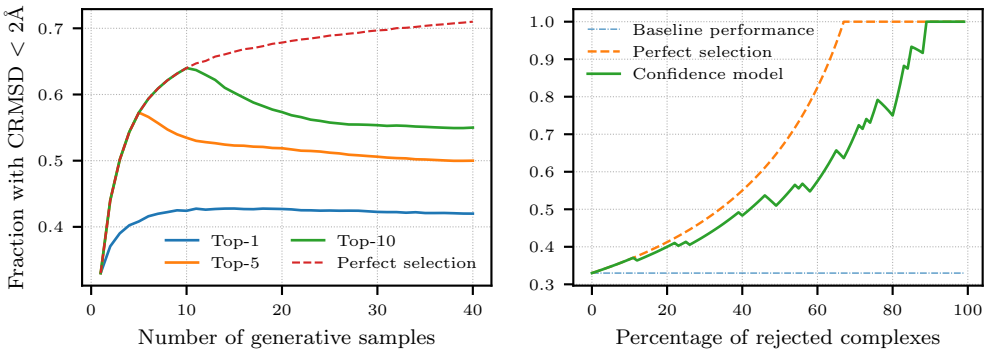


Figure 8: **Left**: Performance of DIFFDOCK-PP for with increasing number of generative samples. Perfect selection refers here to always choosing the sample with the lowest CRMSD. **Right**: Fraction of predictions with CRMSD < 2Å considering only predictions for the part of the dataset where DIFFDOCK-PP is most confident.

5 Conclusion

In this work, we investigated different ways to extend the recent DiffDock [7] and improve its results. In the first part of the report, we explored the datasets that can be used for ligand-protein docking problems. We investigated existing protein-ligand docking datasets, analyzed their curation processes, and evaluated the possibility of duplicating these steps to create our own datasets. Many of these datasets are generated by extracting complexes from the PDB database using various criteria derived from the contents of the PDB structure files [47, 26, 10]. Prominent examples of these datasets include PDBBind and Binding MOAD [47, 26], which seem to be well-suited for ligand-protein docking problems, according to our analysis. However, many of these datasets involve manual verification steps, particularly for the validation of ligands [47, 26, 10], which seems to be a very complicated step to automatize and may require a lot of human resources. Based on these and other reasons discussed in the first part of the report, we concluded that duplicating their curation steps would not be feasible for the scope of this project.
In the first presented contribution of this report, we altered the rigid docking of DiffDock to model receptor flexibility during docking and investigated different directions to incorporate binding pocket information into the diffusion process. We showed that our *pocket-reduction* approach is superior to the existing *pocket-prior* method with accuracy and significant runtime increases. Furthermore, our binding-pocket-conditioned receptor flexibility method matches the performance of the rigid approaches, while showing promising results, especially for docking on unbound structures even at an early training stage with reduced model sizes. This opens the door to predicting the binding pose of a ligand, and also the structural changes in the protein, making DiffDock's method applicable to new areas. However, to confirm the promising results of short training periods, training until convergence with the full model and hyperparameter optimization should be done.
The second presented contribution is DiffDock-PP, a diffusion generative model for rigid protein-protein docking. Our approach is inspired by recent advancements in molecular docking [7], which tackles docking via a generative model over ligand poses. DiffDock-PP outperforms existing deep learning models and performs competitively against search-based methods at a fraction of their computational cost. The effectiveness of our simple approach paves the way for further investigation into deep learning for modeling biomolecular interactions.
All in all, with the methods presented in the report, we were able to explore different applications of diffusion-based docking and expand upon the existing methods.

Acknowledgement

Beyond our mentors, Hannes Stärk and Céline Marquet, we also want to thank Gabriele Corso and Rachel Wu. We are very grateful for all the time they have invested to help us with various problems and have given us insightful input throughout the project. Thank you!

Bibliography

[1] Sankar Basu and Björn Wallner. "DockQ: a quality measure for protein-protein docking models". In: *PloS one* 11.8 (2016), e0161879.

[2] Stephen K Burley et al. "Protein Data Bank (PDB): the single global macromolecular structure archive". In: *Protein crystallography: methods and protocols* (2017), pp. 627–641.

[3] Nanxin Chen et al. "Wavegrad: Estimating gradients for waveform generation". In: *arXiv preprint arXiv:2009.00713* (2020).

[4] Rong Chen, Li Li, and Zhiping Weng. "ZDOCK: an initial-stage protein-docking algorithm". In: *Proteins: Structure, Function, and Bioinformatics* 52.1 (2003), pp. 80–87.

[5] Jordan J Clark et al. "Inherent versus induced protein flexibility: comparisons within and between apo and holo structures". In: *PLoS computational biology* 15.1 (2019), e1006705.

[6] Wikimedia Commons. *Table of amino acids*. 2017. URL: `https://upload.wikimedia.org/wikipedia/commons/a/ac/AAs_table.png`.

[7] Gabriele Corso et al. "DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking". In: *arXiv preprint arXiv:2210.01776* (2022).

[8] Valentin De Bortoli et al. "Riemannian score-based generative modeling". In: *arXiv preprint arXiv:2202.02763* (2022).

[9] Sjoerd J De Vries, Marc Van Dijk, and Alexandre MJJ Bonvin. "The HADDOCK web server for data-driven biomolecular docking". In: *Nature protocols* 5.5 (2010), pp. 883–897.

[10] Jérémy Desaphy et al. "sc-PDB: a 3D-database of ligandable binding sites-10 years on". In: *Nucleic Acids Research* 43.D1 (Oct. 2014), pp. D399–D404. ISSN: 0305-1048. DOI: `10.1093/nar/gku928`. eprint: `https://academic.oup.com/nar/article-pdf/43/D1/D399/17438022/gku928.pdf`. URL: `https://doi.org/10.1093/nar/gku928`.

[11] Israel T Desta et al. "Performance and its limits in rigid body protein-protein docking". In: *Structure* 28.9 (2020), pp. 1071–1081.

[12] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8780–8794.

[13] Richard Evans et al. "Protein complex prediction with AlphaFold-Multimer". In: *BioRxiv* (2021), pp. 2021–10.

[14] Richard A. Friesner et al. "Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy". In: *Journal of Medicinal Chemistry* 47.7 (2004). PMID: 15027865, pp. 1739–1749. DOI: `10.1021/jm0306430`. eprint: `https://doi.org/10.1021/jm0306430`. URL: `https://doi.org/10.1021/jm0306430`.

[15] Octavian-Eugen Ganea et al. "Independent se (3)-equivariant models for end-to-end rigid protein docking". In: *arXiv preprint arXiv:2111.07786* (2021).

[16] Mario Geiger et al. "Euclidean neural networks: e3nn". In: *Zenodo. https://doi.org/10.5281/zenodo* 5292912 (2020).

[17] Hervé Hogues et al. "ProPOSE: Direct Exhaustive Protein-Protein Docking with Side Chain Flexibility". In: *Journal of Chemical Theory and Computation* 14.9 (2018), pp. 4938–4947.

[18] Sheng-You Huang and Xiaoqin Zou. "A knowledge-based scoring function for protein-RNA interactions derived from a statistical mechanics-based iterative method". In: *Nucleic acids research* 42.7 (2014), e55–e55.

[19] Sheng-You Huang and Xiaoqin Zou. "An iterative knowledge-based scoring function for protein–protein recognition". In: *Proteins: Structure, Function, and Bioinformatics* 72.2 (2008), pp. 557–579.

[20] John Ingraham et al. "Illuminating protein space with a programmable generative model". In: *bioRxiv* (2022), pp. 2022–12.

[21] John Ingraham et al. "Illuminating protein space with a programmable generative model". In: *bioRxiv* (2022), pp. 2022–12.

[22] Arian R Jamasb et al. "Deep learning for protein–protein interaction site prediction". In: *Proteomics Data Analysis* (2021), pp. 263–288.

[23] Bowen Jing et al. "Torsional Diffusion for Molecular Conformer Generation". In: *ICLR2022 Machine Learning for Drug Discovery*. 2022. URL: https://openreview.net/forum?id=D9IxPlXPJJS.

[24] Wolfgang Kabsch. "A solution for the best rotation to relate two sets of vectors". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–923.

[25] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[26] Johannes Kirchmair et al. "The Protein Data Bank (PDB), its related services and software tools as key components for in silico guided drug discovery". In: *Journal of medicinal chemistry* 51.22 (2008), pp. 7021–7040.

[27] Dima Kozakov et al. "The ClusPro web server for protein–protein docking". In: *Nature protocols* 12.2 (2017), pp. 255–278.

[28] Marc F Lensink, Raúl Méndez, and Shoshana J Wodak. "Docking and scoring protein complexes: CAPRI 3rd Edition". In: *Proteins: Structure, Function, and Bioinformatics* 69.4 (2007), pp. 704–718.

[29] Jie Liang, Clare Woodward, and Herbert Edelsbrunner. "Anatomy of protein pockets and cavities: Measurement of binding site geometry and implications for ligand design". In: *Protein Science* 7.9 (Sept. 1998), pp. 1884–1897. DOI: 10.1002/pro.5560070905. URL: https://doi.org/10.1002/pro.5560070905.

[30] Zeming Lin et al. "Language models of protein sequences at the scale of evolution enable accurate structure prediction". In: *BioRxiv* (2022).

[31] Zhihai Liu et al. “Forging the basis for developing protein–ligand interaction scoring functions”. In: *Accounts of chemical research* 50.2 (2017), pp. 302–309.

[32] Zhihai Liu et al. “PDB-wide collection of binding data: current status of the PDBbind database”. In: *Bioinformatics* 31.3 (2015), pp. 405–412.

[33] Wei Lu et al. “TANKBind: Trigonometry-Aware Neural NetworKs for Drug-Protein Binding Structure Prediction”. In: *bioRxiv* (2022).

[34] Efrat Mashiach et al. “An integrated suite of fast docking algorithms”. In: *Proteins: Structure, Function, and Bioinformatics* 78.15 (2010), pp. 3197–3204.

[35] Matthew Masters et al. “Deep Learning Model for Flexible and Eflificient Protein-Ligand Docking”. In: *ICLR2022 Machine Learning for Drug Discovery*. 2022. URL: https://openreview.net/forum?id=WNwsnE81meC.

[36] Andrew T McNutt et al. “GNINA 1.0: molecular docking with deep learning”. In: *Journal of Cheminformatics* 13.1 (June 2021), p. 43.

[37] Jens Meiler and David Baker. “ROSETTALIGAND: protein-small molecule docking with full side-chain flexibility”. en. In: *Proteins* 65.3 (Nov. 2006), pp. 538–548.

[38] Rocco Meli et al. “Learning protein-ligand binding affinity with atomic environment vectors”. In: *Journal of Cheminformatics* 13.1 (Aug. 2021), p. 59.

[39] Gautam Mittal et al. “Symbolic music generation with diffusion models”. In: *arXiv preprint arXiv:2103.16091* (2021).

[40] Shuya Nakata, Yoshiharu Mori, and Shigenori Tanaka. “End-to-end protein-ligand complex structure generation with diffusion-based generative models”. In: *bioRxiv* (2022). DOI: 10.1101/2022.12.20.521309. eprint: https://www.biorxiv.org/content/early/2022/12/21/2022.12.20.521309.full.pdf. URL: https://www.biorxiv.org/content/early/2022/12/21/2022.12.20.521309.

[41] Youngshang Pak and Shaomeng Wang. “Application of a Molecular Dynamics Simulation Method with a Generalized Effective Potential to the Flexible Molecular Docking Problems”. In: *The Journal of Physical Chemistry B* 104.2 (2000), pp. 354–359. DOI: 10.1021/jp993073h. eprint: https://doi.org/10.1021/jp993073h. URL: https://doi.org/10.1021/jp993073h.

[42] *PDBBind Official Website*. http://www.pdbbind.org.cn/index.php. Accessed: 2022-02-09.

[43] Vadim Popov et al. “Grad-tts: A diffusion probabilistic model for text-to-speech”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8599–8608.

[44] Emanuele Rodolà et al. “Functional maps representation on product manifolds”. In: *Computer Graphics Forum*. Vol. 38. 1. Wiley Online Library. 2019, pp. 678–689.

[45] Christina EM Schindler et al. “Protein-protein and peptide-protein docking and refinement using ATTRACT in CAPRI”. In: *Proteins: Structure, Function, and Bioinformatics* 85.3 (2017), pp. 391–398.

[46] Dina Schneidman-Duhovny et al. “PatchDock and SymmDock: servers for rigid and symmetric docking”. In: *Nucleic acids research* 33.suppl_2 (2005), W363–W367.

[47] Richard D. Smith et al. “Updates to Binding MOAD (Mother of All Databases): Polypharmacology Tools and Their Utility in Drug Repurposing”. In: *Journal of Molecular Biology* 431.13 (2019). Computation Resources for Molecular Biology, pp. 2423–2433. ISSN: 0022-2836. DOI: https://doi.org/10.1016/j.jmb.2019.05.024. URL: https://www.sciencedirect.com/science/article/pii/S0022283619302967.

[48] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in neural information processing systems* 32 (2019).

[49] Yang Song and Stefano Ermon. “Generative modeling by estimating gradients of the data distribution”. In: *Advances in neural information processing systems* 32 (2019).

[50] Yang Song et al. “Score-based generative modeling through stochastic differential equations”. In: *arXiv preprint arXiv:2011.13456* (2020).

[51] Hannes StÃ¤rk et al. “EquiBind: Geometric Deep Learning for Drug Binding Structure Prediction”. In: (2022). DOI: 10.48550/ARXIV.2202.05146. URL: https://arxiv.org/abs/2202.05146.

[52] Antonia Stank et al. “Protein Binding Pocket Dynamics”. In: *Accounts of Chemical Research* 49.5 (Apr. 2016), pp. 809–815. DOI: 10.1021/acs.accounts.5b00516. URL: https://doi.org/10.1021/acs.accounts.5b00516.

[53] Freyr Sverrisson et al. “Physics-informed deep neural network for rigid-body protein docking”. In: *ICLR2022 Machine Learning for Drug Discovery*. 2022. URL: https://openreview.net/forum?id=5yn5shS6wN.

[54] Simon J Teague. “Implications of protein flexibility for drug discovery”. en. In: *Nat Rev Drug Discov* 2.7 (July 2003), pp. 527–541.

[55] Nathaniel Thomas et al. “Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds”. In: *arXiv preprint arXiv:1802.08219* (2018).

[56] René Thomsen and Mikael H. Christensen. “MolDock: A New Technique for High-Accuracy Molecular Docking”. In: *Journal of Medicinal Chemistry* 49.11 (2006). PMID: 16722650, pp. 3315–3321. DOI: 10.1021/jm051197e. eprint: https://doi.org/10.1021/jm051197e. URL: https://doi.org/10.1021/jm051197e.

[57] Raphael Townshend et al. “End-to-end learning on 3d protein structure for interface prediction”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[58] Oleg Trott and Arthur J Olson. “AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading”. en. In: *J. Comput. Chem.* 31.2 (Jan. 2010), pp. 455–461.

[59] Ilya A Vakser. “Protein-protein docking: From interaction to interactome”. In: *Biophysical journal* 107.8 (2014), pp. 1785–1793.

[60] Sjoerd J de Vries et al. “A web interface for easy flexible protein-protein docking with ATTRACT”. In: *Biophysical journal* 108.3 (2015), pp. 462–465.

[61] A. Wojtczak et al. *TRANSTHYRETIN (ALSO CALLED PREALBUMIN) COMPLEX WITH THYROXINE (T4)*. Apr. 1997. DOI: 10.2210/pdb2rox/pdb. URL: https://doi.org/10.2210/pdb2rox/pdb.

[62] Yumeng Yan et al. "HDOCK: a web server for protein–protein and protein–DNA/RNA docking based on a hybrid strategy". In: *Nucleic acids research* 45.W1 (2017), W365–W373.

[63] Yumeng Yan et al. "The HDOCK server for integrated protein–protein docking". In: *Nature protocols* 15.5 (2020), pp. 1829–1852.

[64] M. Yogavel et al. *Crystal structure of PfKRS complexed with chromone inhibitor*. Mar. 2019. DOI: 10.2210/pdb6agt/pdb. URL: https://doi.org/10.2210/pdb6agt/pdb.

[65] Yong Zhao and Michel F Sanner. "Protein–ligand docking with multiple flexible side chains". In: *Journal of Computer-Aided Molecular Design* 22.9 (Sept. 2008), pp. 673–679.

Appendix

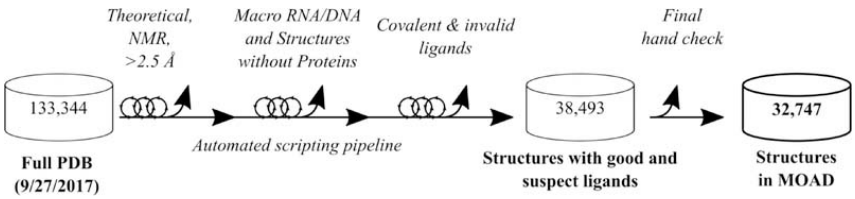


Figure 9: Extraction steps of Binding MOAD dataset. This figure was taken from [47].

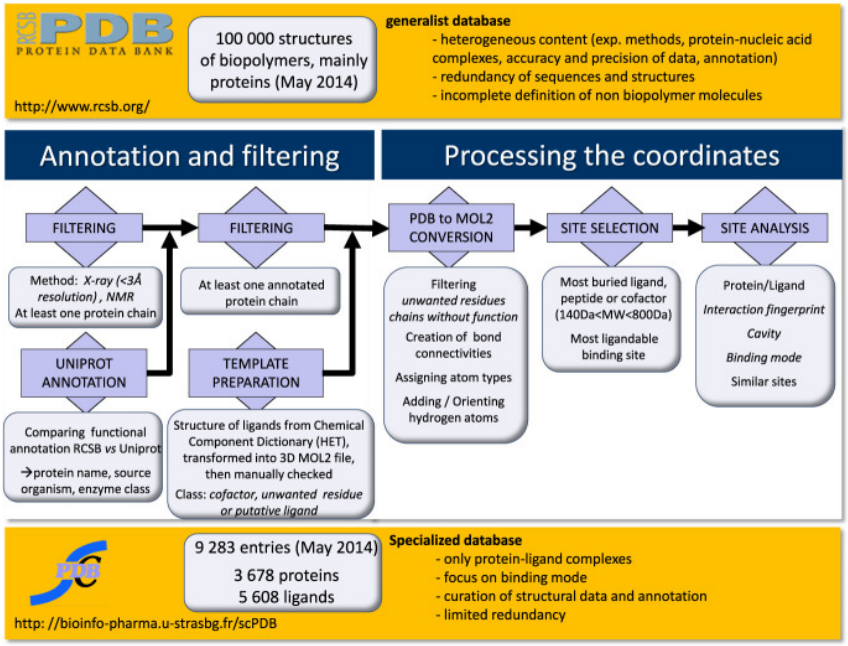
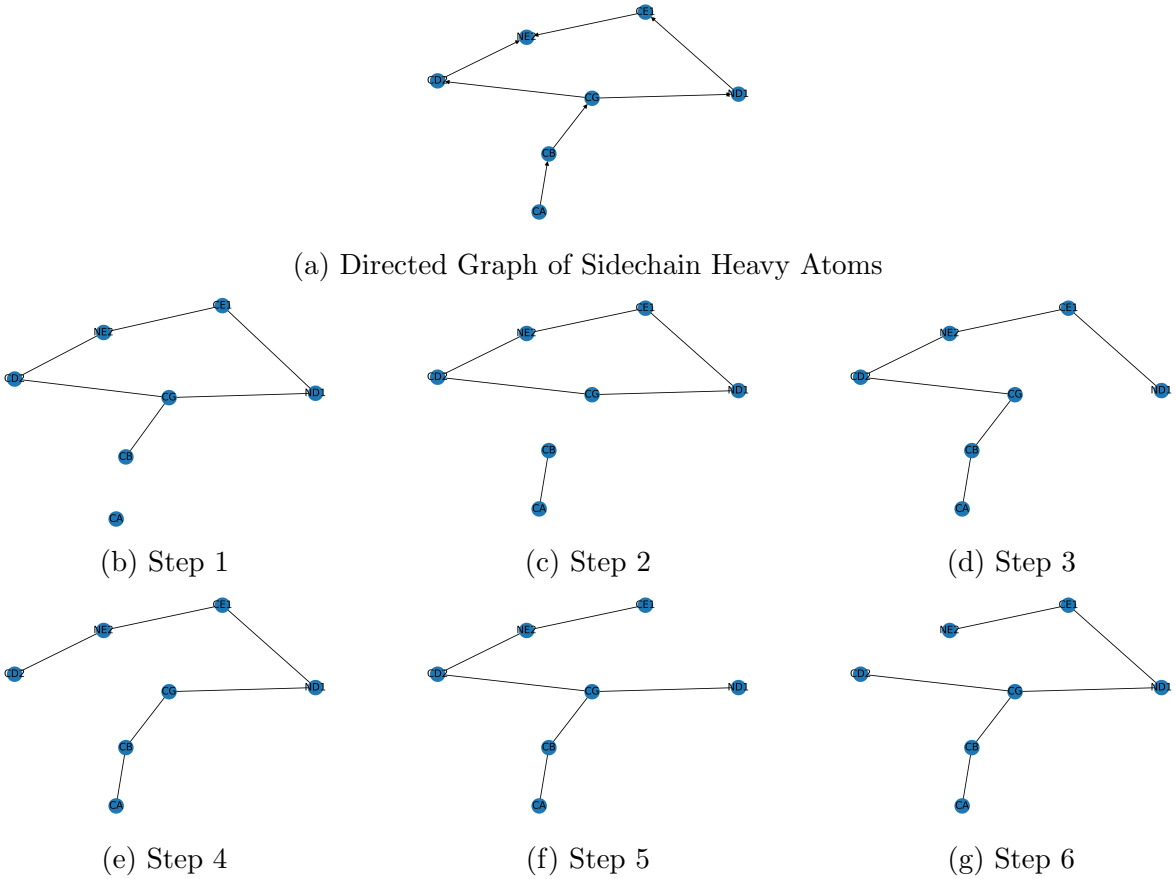


Figure 10: Extraction steps of scPDB dataset. This figure was taken from [10].



(a) Directed Graph of Sidechain Heavy Atoms

(b) Step 1

(c) Step 2

(d) Step 3

(e) Step 4

(f) Step 5

(g) Step 6

Figure 11: Graph Traversal to compute rotatable bonds for Histidine, algorithm 1

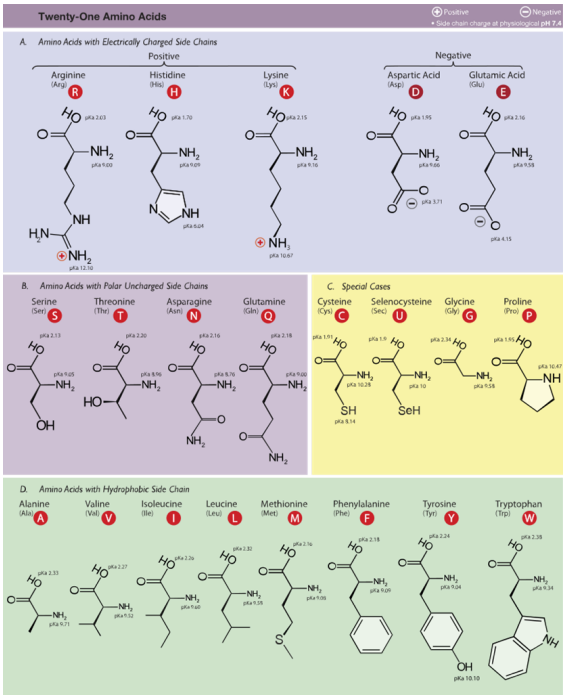


Figure 12: Table of possible Sidechains — taken from [6]

Algorithm 1: Graph Traversal to compute rotatable bonds

Input: Atom positions x, atom names $\mathcal{N}$
Output: Rotable bonds $\mathcal{B}$, rotation mask $\mathcal{M}$
$(x, \mathcal{N}) \leftarrow \texttt{removeHydrogens}(x, \mathcal{N})$;
$G \leftarrow \texttt{constructDirectedGraph}(x, \mathcal{N})$;
for $e \in edges(\texttt{BFS}(G))$ **do**
  $G_U \leftarrow \texttt{toUndirected}(G)$;
  $G_U \leftarrow \texttt{removeEdge}(G_U, e)$;
  if *not* $\texttt{isConnected}(G_U)$ **then**
    $c \leftarrow \texttt{connectedComponents}(G_U)$;
    if $\texttt{size}(sorted(c)[0]) > 1$ **then**
      $\mathcal{M}.append(c[1])$;
      $\mathcal{B}.append(e)$;
    end
  end
end

Table 4: Model sizes compared to the DiffDock [7] model.

PARAMETER	DiffDock	Pocket-Reduced	Flexible Sidechains
CONVOLUTION LAYERS	6	5	4
NUMBER OF SCALAR FEATURES	48	25	20
NUMBER OF VECTOR FEATURES	10	6	5
NUMBER OF PARAMETERS (M)	**20.24**	**2.53**	**2.20**