Virtual Screening with 3D Molecular Fingerprints

Rory Bedford*

MSc Machine Learning

Andrea Karlová,
Prof. Brooks Paige

Submission date: 19 September 2022

Acknowledgments

First and foremost I would like to thank Andrea Karlová for her wonderful supervision throughout this project. She has directed me towards an incredibly stimulating field and has provided invaluable guidance at all stages of this work.

I would also like to thank Brooks Paige for acting as internal supervisor. He has given us fantastic advice on exciting directions we can take the project in, and has provided thoughtful insights on many of our results.

I would like to extend my thanks to everyone at April19 Discovery Inc. for letting me collaborate with them. In particular, I would like to thank Wim Dehaen for his seemingly endless knowledge of computational chemisty and bioinformatics, and Andrei Penciu for his contributions to our reading group and computational help.

I would also like to thank Jude Wells for our stimulating library discussions about our work, and for helping me keep up-to-date with the latest research in the field.

Additionally, I want to express my heartfelt thanks to all of my coursemates for providing such a supportive and sociable work environment, and to all of my flatmates for providing a loving and peaceful home environment. You've all helped make this a truly enjoyable year for me and I am deeply grateful to you all.

Finally, I want to thank my parents for their endless support and unconditional love.

Abstract

Protein-ligand docking is a crucial component of virtual screening, and plays a central role in a computationally-driven drug discovery pipeline. Classical docking methods are slow, and therefore create a bottleneck in the drug discovery process, limiting the number of compounds that can be screened. Modern machine learning (ML) models provide a dramatic computational speedup compared to these classical methods. However, these models currently present problems that limit their usability.

A major task in the development of such a ML model is how to represent the molecules in a way that captures all the relevant information about them, such as the atoms present, their topology, and 3D structure. Recent advancements in geometric deep learning (GDL) have allowed a far wider variety of mathematical objects to be used as molecular representations. These models then implicitly learn molecular fingerprints in their layers, that compress the relevant information and can be reused in other prediction tasks.

EquiBind [50] is a GDL-based docking model which builds SE(3) equivariance into its architecture, and promises to find the optimal docking pose for any input ligand. In this thesis, we question this assumption, and show that EquiBind has a high degree of dependence on the input ligand conformation. We highlight a number of challenges involved in using EquiBind for virtual screening, and show a large degree of variability in its performance across different targets.

Formally, we present two hypotheses:

Hypothesis 1: EquiBind has comparable performance to classical docking methods on targets in its training set, but is not a useful docking method on unseen targets.

Hypothesis 2: EquiBind learns implicit 3D fingerprints in its layers that can be generalised to other prediction tasks, including binding affinity prediction.

In particular, we present three redocking experiments, in which we compare EquiBind's performance with AutoDock Vina [16]. First of all, we redock a set of experimentally determined crystal ligand poses on targets in EquiBind's training set, then test set. In a virtual screening context, we would not know the output ligand conformation, so we then perform another experiment in which we generate input conformations for the ligands and redock them. Finally we examine EquiBind's behaviour on an unseen target - MurD ligase - in a realistic drug-discovery pipeline. We conclude that the performance of EquiBind is overstated due to its performance in these experiments. However, we believe a larger, more diverse training set could improve on a number of these drawbacks.

Additionally, we construct EquiBind-score, a novel model trained to predict binding affinity, which uses hidden layers from EquiBind as 3D molecular fingerprints. This model has impressive performance on its test set, composed of complexes from PDBbind [32]. However, this model struggles to differentiate actives from decoys.

Contents

Chapter 1

Introduction

In this chapter we introduce the field of drug discovery and discuss the motivation behind the research undertaken in this thesis. We then provide some context for the topic of virtual screening with a brief overview of the entire drug discovery pipeline. Next we examine two important components of the virtual screening process: docking and affinity prediction. We discuss the classical* computational solutions to these problems, and their shortcomings. We then formally define our research hypothesis and give a brief description of the structure of the remainder of the thesis.

1.1 Motivation

A formal definition of a drug, given by the FDA, is as follows:

A substance (other than food) intended to affect the structure or any function of the body.

In this thesis we consider only small molecule drugs, known as pharmaceuticals, which are organic chemicals with a mass less than 1000Da. These drugs work by binding to a target in the body. Drug targets are typically much larger organic molecules, the most common targets being proteins, although other targets such as RNA do exist which we do not discuss in this thesis. Proteins can be thought of as biological machines that carry out specific functions in the body, such as enzymes that catalyse reactions, or structural proteins that make up a cell wall. When a drug binds to a protein it affects its function, which is how the drug has its desired therapeutic effect.

Historically, drugs were discovered either by chance, or through investigation of natural medicines to determine their active compounds. In fact, the discovery that plant medicines work through the specific action of one or more active compounds at the molecular level led to a revolution in the field of pharmacology, and created a shift away from natural remedies being used as medicines towards modern pharmaceuticals which often have just one active compound. The story of the field of pharmacology is one of increasing intentionality behind the process of drug discovery. While in the past we were subject to simply using the medicines that nature has provided us with, leaving the vast majority of the chemical space untapped, increasingly we are able to target a wide variety of diseases through the intentional discovery of new drugs. However, this shift has not yet reached its conclusion, in that modern drug discovery still involves large elements of chance. The field is still evolving rapidly, and we believe that ML based approaches will play a large part in aiding drug design in the future.

*Here 'classical' methods refer to computational methods that do not involve any machine learning.

Drug discovery is an expensive and time consuming process. Current estimates place the cost of discovering a single drug at around US$1.8 billion, with a timeline of around 10 years of development before regulatory approval [41]. A significant reduction in both the temporal and monetary cost associated with drug development would be of immense benefit to medicine: it would allow us to treat novel diseases more rapidly in an epidemic; or treat rare diseases which lack the funding to develop treatments given current costs.

Virtual screening is one component of the drug discovery pipeline in which machine learning (ML) approaches show potential for considerable computational speedups, in some cases a thousand-fold speedup [50], when compared to classical computational methods. Virtual screening is the process of computationally analysing large libraries of small molecules to determine which molecules may have the desired properties, such as binding to the target, in order to reduce the number of candidate molecules to a manageable size.

The chemical space is enormous: estimates place the number of possible drug-like molecules at around 10^{60} [7]. In a virtual screening scenario, researchers may be faced with a library of over a billion molecules that needs to be filtered down to just a few thousand hits (promising drug candidates) for further experimentation. This is why efficient algorithms for hit identification are needed.

In particular, the work undertaken in this thesis is motivated by an open source antibiotics challenge, in which fragment-based lead discovery methods are used to search for an inhibitor of the MurD Ligase, an enzyme present in many bacteria which is crucial for their survival. A small team of researchers from UCL Computer Science and April19 Discovery Inc. presented a set of hits for synthesis and experimentation using a modern ML approach. As a small part of this work we performed some experiments assessing the suitability of the ML based docking method EquiBind [50] in this pipeline, which we present in Section 4.3. For details about this project see: https://github.com/opensourceantibiotics/murligase.

1.2 The Drug Discovery Pipeline

Drug discovery is an immensely challenging problem for a multitude of reasons. The causes of disease and their potential routes of treatment are varied and often tricky to identify. The method of action of a drug, and in particular whether it binds to a target, is incredibly difficult to determine, and the solution needs to draw upon experimental biochemistry, computational chemistry, and other fields. Furthermore, drugs may act in unpredictable ways in humans leading to side effects and toxicity.

The modern pipeline can be split into four distinct phases:

Drug discovery pipeline:

Phase 1: *The early drug discovery process*, in which a target for the drug is identified, and lead molecules (promising drug candidates) are developed and tested.

Phase 2: *The pre-clinical phase*, in which the drug is tested in the lab and in animals. The purpose of this stage is to assess the drug's efficacy, toxicity, and dosages for clinical trials.

Phase 3: *Clinical trials*, in which the drug is tested on increasingly large groups of humans. The purpose here is to continue to assess safety concerns, such as side effects, and to test the drug's efficacy.

Phase 4: *Regulatory approval* and ongoing monitoring.

While there is room for improvements in efficiency in all these phases, in this thesis we are concerned with phase 1: the early drug discovery process, which we will discuss in more detail.

There are a number of different approaches in the field of early drug discovery depending on factors including: computational and experimental resources available to the researchers, knowledge about the target including whether drugs for it already exist, and what type of library of compounds the researchers have to work with. This makes it difficult to give a concise overview of the subject. We split the process into four distinct phases [22]. However, this linear description of the process is quite a crude approximation to the truth: often earlier phases are revisited, informed by the results of later phases.

Early drug discovery process:

Phase 1: *Target identification and validation*, in which a target for the drug (usually a protein) is put forward by researchers. Examples of targets include a protein necessary for the survival of a virus, or an overexpressed protein in a cancer cell. Note that some proteins have more than one pocket (a viable place for a drug to bind), and typically only binding in one pocket will give the desired therapeutic effects, hence only that pocket is considered to be the target.

Phase 2: *Compound screening*, in which hits that bind to the target are discovered. There are a number of ways of discovering hits, however, typically a top-down approach is employed in which researchers start with a large library of full-size compounds (weights around 500Da) which is filtered down. Two common methods for this are high-throughput screening (HTS) and virtual screening.

HTS methods use robotics, detectors and software to experimentally conduct millions of assays which can determine which drugs are active. Alternatively, virtual screening uses computational methods including docking and pharmacophore models to estimate which drugs are active. We discuss virtual screening in more detail in Section 1.3. Virtual screening can save on cost, time and resources as HTS is a complicated and expensive process. However, this may be at the expense of accuracy. Often virtual screening and HTS are used in series - for example, virtual screening may be employed first to reduce a large library down to a manageable size for HTS.

An alternative to the top-down approach is a bottom-up approach, such as fragment-based lead discovery (FBLD). In FBLD small libraries of small molecules (weights around 200Da) are screened to find hits which weakly bind, and these hits are then grown or combined to produce hits with a higher binding affinity.

In both approaches, researchers will typically still be left with potentially hundreds or thousands of hits. A number of approaches are employed to filter these hits into leads for further experimentation. Examples include heuristic filters such as the Lipinski Rule of Five [31], which gives simple guidelines to estimate whether a drug is orally active. Hits can also be grown in this stage to increase binding affinity further. Hits are sometimes clustered based on structure, and only one representative of each cluster is put forward as a lead.

Phase 3: *Secondary assays*, in which further experimental testing is done on the leads. For example, experiments are performed to see whether the drug works within a cell. The half-maximal inhibitory concentration (IC50), which we discuss in Section 1.3.1, can be

measured, as well as the entire dose-response curve, the structure-activity relationship between the drug and the target, and a host of other important results.

Next: Lead compounds are sent off for preclinical testing.

It is worth noting that the entire process can fail at any stage, even after millions of dollars have been invested in research. A promising lead could turn out to be toxic in humans, or a target may turn out to not be druggable, for example. This is part of the reason why drug discovery can be such an expensive and time consuming process.

1.3 Virtual Screening

As discussed in Section 1.2, virtual screening is the process of computationally scanning a large library of drug-like molecules against a target to identify hits. There are two main paradigms: structure-based and ligand-based.

Structure-based virtual screening uses knowledge of the structure of the target, along with models of the possible chemical interactions between the ligand and the target, to model their interactions and identify hits. Docking and affinity prediction are important components of structure-based virtual screening, which we discuss in detail in Section 1.3.1, as they form the basis for much of the work in this thesis.

Ligand-based virtual screening can be used when we already have a set of ligands that are known to bind to the receptor. Ligand-based methods use information about these pre-existing hits to screen databases for other potential hits. A handful of different methods exist for ligand-based screening, however, we will briefly discuss only one particular method, called a pharmacophore model, because of its use in our pipeline for the antibiotics challenge.

A pharmacophore model abstracts features present in all known active compounds, such as a hydrogen donor at a certain location, or an aromatic ring at a different location, and then makes the assumption that these features must be present in all unknown hits, and so can use them as a filter. This is a reasonable assumption to make because, in the example of the conserved hydrogen donor, it is likely that the target and the drug are forming a hydrogen bond at that location, and other drugs are likely to require this same hydrogen bond to bind to that pocket. The model needs to run over all low-energy conformers of the known actives to choose their binding conformations - a concept we discuss in Section 1.3.1.

1.3.1 Docking

Docking is the task of computationally predicting the relative orientations of two bound molecules. The molecules can be of many different types: examples include proteins docking to other proteins, or drug-like molecules docking to proteins or nucleic acids. A wide range of organic molecular complexes are found in biochemistry and any of these can be docked. In this thesis, we are concerned only with small drug-like molecules docking to proteins.

Throughout the remainder of this thesis we may use the terminology of docking, in which the larger molecule, in our case a protein, is referred to as the 'target', or alternatively the 'receptor', and the smaller molecule, here the drug, is referred to as the 'ligand'. Furthermore, the 'pocket' refers to the location within the target where the ligand docks.

A target will only have a small number of pockets, usually just one. To each pocket a large number of different ligands may be known to bind. Often a pocket is a concave part of the

protein surface, with a shape such that certain ligands fit into it, hence the name. The lock-and-key analogy likens the target to a lock and the ligand to a key shaped perfectly to fit the target's pocket, as depicted in Figure 1.1.

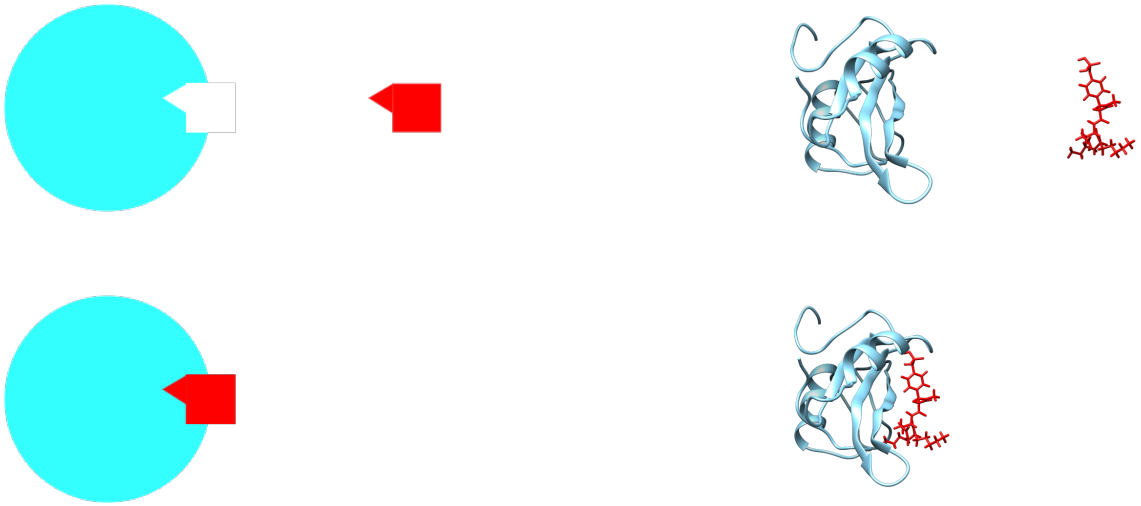


Figure 1.1: Lock-and-key analogy for docking

It is not just the ligand and pocket shapes that matter, but also a number of chemical properties. When two molecules dock, a number of different types of chemical interactions can occur between them, including electrostatic interactions, hydrogen bonds, and van der Waals interactions. These interactions are what hold the two molecules together in a stable compound. In different docking positions these interactions may be weaker or stronger - for example, rotating or translating the ligand could lead to a negatively charged ion in the ligand moving further away from a positively charged ion in the target, reducing the strength of the electrostatic interaction between them. The optimal docking position will be the position in which the sum of all interactions is strongest, forming the most stable compound. This is the docking position we seek to find as it is the position nature will favour.

Docking software requires the structure of the target and the ligand as inputs. By 'structure' we mean the three dimensional shape, which computationally means a list of coordinates of all atoms present in the molecule along with their atomic type. We discuss the file types used for structural information in Chapter 2. The software then estimates the optimal structure of the bound compound and returns this as a structure file.

There are a few different types of docking. First, we will discuss the difference between 'flexible' and 'rigid' docking. When two molecules bind to from a compound, the structures of the individual molecules may change. For example, a protein may bend or twist slightly to accommodate a ligand in its pocket. Docking software can model this flexibility, in which case it's called flexible docking, or it may keep the structures fixed while just rotating and translating the molecules, in which case it's called rigid docking. Some docking software also models only ligand flexibility while keeping the target rigid. Trying to model flexibility can make the problem much more difficult, as it greatly increases the number of degrees of freedom involved. In some compounds, experimental data shows that neither molecules' structure changes much when they bind, in which case rigid docking can be used effectively. This is not always the case, however, so expertise is needed to determine which to use. Many docking software packages can perform both rigid and flexible docking.

It is worth discussing ligand flexibility in more depth. Proteins typically have a single, well defined unbound structure, that can be measured experimentally through x-ray crystallography,

or estimated computationally with a method such as AlphaFold. Proteins can misfold but this is the exception that proves the rule. Small drug-like molecules on the other hand typically do not have a single preferred unbound structure but rather a whole set of stereoisomers, which are molecules sharing the same molecular formula and bonds but with different structures. It turns out that the large degree of ligand flexibility is almost entirely due to rotations about single bonds. Molecular structures that differ only by rotations about single bonds are known as different conformations. Single bonds find it far easier to rotate than to stretch or shrink, and hence we often model ligand flexibility only through different conformations. In the unbound state, not all conformations are equally likely. Interatomic interactions affect which conformations are more likely to occur. These include electrostatic interactions and steric effects, the latter of which describe the tendency for atoms to avoid overlap. These interactions together contribute to the free energy of the conformation. While unbound the molecule will be in one of many local minima of its free energy, known as conformers. An example of two conformers is shown in Figure 1.2. When the ligand binds to a target, its conformation will change to fit the geometric and chemical properties of the pocket.

Figure 1.2: Two different conformers of the same molecule

We will now discuss the difference between 'blind' docking and 'site-specific' docking. In many cases, the binding pocket to which we want to dock is already known to us. This is often the case when we have a well studied target. Many different ligands may already be known to bind to a single pocket in this target, for example. In this case we can specify the pocket to the docking software, which can greatly reduce the computation time as only a fraction of the entire protein surface needs to be checked. In other cases, however, we need to use blind docking, which is where we don't specify the pocket. This may be necessary in cases where we have a brand new target with little information known about it.

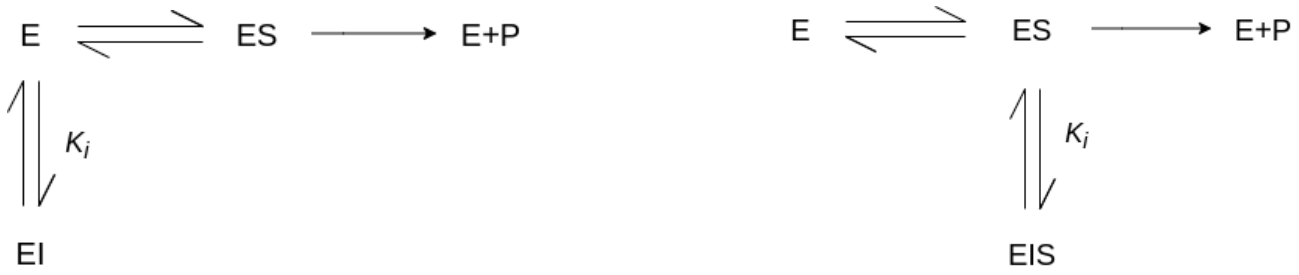
We will now discuss the problem of binding affinity prediction, sometimes referred to as 'scoring'. Binding affinity describes how well the ligand binds to the target. Generally, if the sum of the attractive forces between the ligand and target is high, they will have a high binding affinity, or equivalently, if the complex has a low free energy relative to the unbound molecules. If the ligand and target do not bind to form a stable complex then it makes no sense to speak of their binding affinity, which does not exist.

There are two closely related measures of binding affinity - the dissociation constant (K_D), and the inhibition constant (K_i). Additionally, the half maximal inhibitory concentration (IC_{50}) is sometimes used, which is not a binding affinity but rather a proxy for binding affinity. We now review these three measures.

The dissociation constant [30] describes the ratio of bound to unbound protein-ligand complexes in equilibrium. In formal notation:

$$\mathrm{L} + \mathrm{P} \leftrightharpoons \mathrm{LP}$$
$$\mathrm{K}_D = \frac{[\mathrm{L}][\mathrm{P}]}{[\mathrm{LP}]}$$

Where L represents the ligand, P the protein, $\leftrightharpoons$ refers to the state where a dynamic chemical equilibrium is reached for a reversible process, such as complexation and decomplexation, and the square brackets give the molar concentration of the enclosed molecule. K_D has units of mol/L, which is often called a molar (M).



(a) Competitive inhibition

(b) Non-competitive inhibition

Figure 1.3: Two different pathways for enzyme inhibition

The inhibition constant K_i [14] is defined for enzyme-inhibitor systems, in which an inhibitor, typically a small molecule, binds to an enzyme, blocking its activity. There are two types of inhibition: competitive and non-competitive, and the definition of the inhibition constant depends on the type. Competitive inhibition occurs when the inhibitor binds to the enzyme directly blocking the substrate from binding. Non-competitive inhibition occurs when the inhibitor can bind to the enzyme-substrate complex, but still reduces the activity of the enzyme. Figure 1.3 shows both inhibition pathways, where E represents the enzyme, I the inhibitor, S the substrate (the molecule(s) the enzyme acts on), and P the product of the reaction catalysed by the enzyme.

We then define K_i as follows. In the competitive case:

$$\mathrm{K}_i = \frac{[\mathrm{E}][\mathrm{I}]}{[\mathrm{EI}]}$$

And in the non-competitive case:

$$\mathrm{K}_i = \frac{[\mathrm{ES}][\mathrm{I}]}{[\mathrm{ESI}]}$$

K_i therefore also has units of M. Note that this is not the complete picture. Enzyme inhibitor pathways can get more complex than this. Furthermore, the inhibition constant is sometimes used for systems that are not even enzyme inhibitors. For a more complete description we refer the reader to Rossotti and Rossotti [44].

Finally, we have the half maximal inhibitory concentration (IC_{50}) [51]. This measures how well a molecule inhibits a particular biological process. Specifically, it is the concentration at which that particular biological function is inhibited by 50% in vitro. This biological function could again be an enzyme that catalyses a reaction, or a receptor involved in a signalling

pathway, for example. IC_{50} also has units of M. It does not directly measure binding affinity, but can be converted to an affinity via the Cheng-Prusoff equation [12].

Binding affinity prediction refers to the computational prediction of one of the above experimentally determined measures of binding affinity for some protein-ligand complex. Often K_i and K_D are treated as interchangeable for this problem, as in Boyles, Deane, and Morris [8], in which case we can simply refer to the binding affinity as K. Often we alternatively will predict the negative base-10 logarithm of K, denoted pK. The reason for this is that the range of binding affinities can be very broad, so the logarithm compresses this. Additionally, the experimental error in measuring binding affinity can be very broad - sometimes experimental results can differ by an order of magnitude - and so taking the logarithm also compresses this error.

$$\mathrm{pK} = -\log_{10} \mathrm{K}$$

Note that a lower K_D, K_i, or IC_{50} corresponds to a higher binding affinity, or equivalently the complex has a lower energy. A higher pK corresponds to a higher binding affinity.

1.3.2 Classical Methods

We now give an overview of the existing classical computational techniques for docking.

Docking algorithms are typically comprised of two parts: a search algorithm, and a scoring function. The overarching idea of classical docking methods is to sample a docked pose, estimate how good it is, and then use this information to sample another pose. With good computing power thousands or millions of poses can be sampled, and the best poses are returned by the algorithm. If enough poses are sampled, these returned poses should in theory approximate the physical solution very closely.

The scoring function gives an approximate prediction of the binding affinity of a given docked pose. The search algorithm determines which poses to sample, and uses information about previous poses and their scores to do this. We now discuss these two components in more detail.

Mathematically, we can consider each pose to be a point on a high dimensional manifold, called the search space. Each dimension corresponds to a degree of freedom. In rigid docking, there are only six degrees of freedom - three translations and three rotations of the ligand (assuming we keep the receptor fixed), corresponding to the SE(3) group. Additionally, we can incorporate ligand flexibility. As discussed in Section 1.3.1, often we will only model ligand flexibility through different ligand conformations, which adds a single degree of freedom for each rotatable bond in the ligand. Due to the size of a typical receptor, including receptor flexibility in a similar way drastically increases the number of degrees of freedom making the problem much harder. Hence modelling receptor flexibility is still very much an open issue. As discussed before, it is sometimes okay to treat the receptor as rigid, but in some cases this give poor results.

The scoring function assigns a scalar field to the search space - that is, it gives a score which can be calculated for every possible pose. There are a number of different scoring functions. Sotriffer et al. [49] divide these into three main categories: force field, empirical, and knowledge-based, which we now outline. We additionally mention ML-based scoring functions.

Scoring functions:

Force field: Force field methods are typically based on classical mechanics: they try to account for all forces between and within the ligand and receptor, including bond-lengthening forces, electrostatic forces, and van der Waals interactions. Based on the sum of these terms they estimate a free energy for each pose.

Empirical: Empirical scoring functions essentially count the number of interactions between the ligand and receptor based on which atoms are sufficiently close. Each interaction type is given a score, which is calculated through the statistical analysis of experimental data. Interaction types include hydrogen bonds, ionic interactions, and surface complementarity terms.

Knowledge-based: Knowledge-based methods also statistically analyse experimental structural datasets, counting the number of close intermolecular contacts. Unlike empirical methods, they do not make use of binding affinity data, but instead are based on the assumption that contacts that are seen more often than random must be energetically favourable. These methods work well for docking but cannot give an accurate affinity.

Machine-learning: Supervised learning methods are similar in this context to the empirical methods described above in that they learn from pre-existing databases. However, they can be more powerful since they learn the functional form mapping the structure to an affinity instead of this being assumed.

We briefly note that the non-ML scoring functions only approximate the binding affinity. They do this well enough to rank different poses relative to each other, and are therefore useful for docking, but do not output an accurate affinity that can be compared with experimental results. Only ML-based scoring functions have achieved accuracy capable of this [11].

With a score assigned to every pose, the goal of the search algorithm is to find the pose with the best score (the highest binding affinity or lowest energy). This corresponds to finding the global minimum of the scalar field on our manifold. The problem is complicated by the size of proteins, the number of degrees of freedom, and the existence of many local minima. Site-specific docking aids this first problem by assigning a box around the protein pocket from which poses are sampled. We discuss five types of search algorithms: naïve search, gradient methods, genetic algorithms, shape-complementarity methods, and molecular dynamics simulations. The first two methods are mentioned for completeness but are ineffective and hence not used in any docking software.

Search algorithms:

Naïve search: Naïve search methods try to sample as much of the search space as possible, either through uniform samples of conformations and poses, or through a large number of stochastic poses. These methods obviously fail for tricky problems since limits on compute resources means it is impossible to sample widely from the entire search space.

Gradient methods: Gradient methods use gradient ascent/descent to find maxima/minima. These methods typically fail due to the extremely high number of local minima/-maxima.

Genetic algorithms: Genetic algorithms mimic the process of natural selection. Starting from a randomly generated initial population of poses, the poses are scored, and a set of the best poses are selected to breed the next generation. Breeding methods include crossover, in which traits of selected poses are combined to create new poses, and mutation, in which noise is added to the poses. Once a large sample is generated, only the best poses from this next generation are selected and the process repeats. When parameterised

well, genetic algorithms work very well for docking. *AutoDock* [37] is a notable example of docking software that uses a genetic algorithm.

Shape-complementarity methods: Shape-complementarity methods search for similarities between the receptor and ligand, including geometric similarities and chemical matches that could indicate the presence of a bond such as a hydrogen bond. The underlying premise follows the lock and key analogy - that is, the ligand should fit well into the pocket. A notable example of docking software taking this approach is *DOCK* [28]. Note these methods do not use a scoring function, however, they are often used in conjunction with other methods that do. For example, shape-complementarity methods are often used to generate starting poses for score optimisation methods.

Molecular dynamics simulations: Molecular dynamics is an entire field of study which performs computational simulations of molecules based on Newtonian mechanics, with numerous applications in material sciences, biophysics and other fields. These methods can be applied to docking with good accuracy, but are very computationally intensive. We will not discuss these methods in detail as they deserve a lot more attention but refer the reader to Singh, Bani Baker, and Singh [48].

It must be noted that we have given an incredibly brief overview here of what is a very large field of study. The reality of docking is much more complex than what we have presented. For example, many docking algorithms will use several different scoring functions together. They may use different search algorithms in conjunction with each other. There are also many other techniques we haven't had time to mention. For a comprehensive overview of the state-of-the art of the field, we refer the reader to Morris and Lim-Wilby [36].

1.3.3 Performance of Classical Methods

Having discussed the techniques used for classical docking, in this section we give present a brief overview of the performance of classical docking methods. In particular, we look at the performance in terms of accuracy of the docked pose, but also of the average computational time taken to predict each pose, due to the importance of speed when using docking software for virtual screening.

The typical metric used to evaluate the quality of a docked pose is the root mean square deviation (RMSD). To calculate the RMSD, simply calculate the absolute square displacement between each atom in the predicted pose vs its corresponding atom in the experimental pose, then take the mean of these then root the answer.

We now quote some results from Cuzzolin et al. [13], a paper which benchmarks and compares some of the most popular docking algorithms. This is a challenging task, complicated by the fact that different docking algorithms have a number of different settings which are hard to align to make a fair comparison. Additionally, different software is optimised for different hardware: for example, some docking software more efficiently uses multi-core CPUs than others. Additionally, some are optimised for GPUs while others are not. This makes benchmarking a slightly controversial topic within the field.

Cuzzolin et al. [13] performed their benchmarking tests on the target human checkpoint kinase 1 (h-Chk1), with 20 different ligands. We note that this is quite a small dataset so we expect these results to be of low quality. The evaluated the following docking algorithms: AutoDock [37], AutoDock Vina [16], Glide [17], GOLD [23], MOE [55], PLANTS [27], and rDock [45]. Their computations were run on a 64-core CPU.

We now quote some results from this benchmarking study. Most of the docking algorithms

performed with a ligand RMSD of between 4Å and 12Å. We do not go into these results in depth due to the small dataset size - for a more rigourous assessment of the accuracy of different docking methods see Huang [21]. Of particular interest to us is the runtime of each method, which we show in Table 1.1.

Docking Method	Average runtime per complex/s
AUTODOCK-ga	973.5
AUTODOCK-lga	633.3
AUTODOCK-ls	7.45
GLIDE-sp	46.8
GOLD-asp	133.4
GOLD-chemscore	136.2
GOLD-goldscore	401.7
GOLD-plp	98.6
PLANTS-chemplp	61.4
PLANTS-plp	23.3
PLANTS-plp95	16.6
MOE-affinitydg	17.6
MOE-londondg	18.4
MOE-gbviwsa	131.9
RDOCK-std	20.0
RDOCK-solv	31.9
VINA-std	132.7

Table 1.1: Average runtime per complex [13]

While classical docking software provides moderately good poses, its runtime is very slow. In a virtual screening context, where we may want to screen hundreds of thousands of molecules, runtimes of order around 100s are incredibly restrictive. Obviously one solution is to distribute the process across, say, a large cluster, but this is incredibly expensive. Therefore, in order to perform cheaper screens on far larger datasets, new techniques are needed that can greatly reduce the runtime needed for docking. This is where machine learning (ML) based approaches show incredible promise.

Chapter 2

Datasets

2.1 PDBbind

The Protein Data Bank (PDB) [5] is a database containing structural information for a large number of biological molecules. The molecules included are predominantly proteins or nucleic acids, but it also includes DNA, carbohydrates, and a number of complexes. The structural information is experimentally determined, typically through either X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, or cryo-electron microscopy. References are provided for the work done to determine each structure. Due to the role of crystallisation in X-ray crystallography these experimental structures are often referred to as crystal structures.

PDBbind [32] is a refined version of PDB containing only molecular complexes, along with their experimentally determined binding affinities (either IC_{50}, K_i, or K_D). The complexes included in the latest version of PDBbind (v.2020) are protein-ligand, protein-protein, protein-nucleic acid, and nucleic acid-ligand. There are 23,496 complexes in PDBbind compared to 157,974 in PDB. Details of the numbers of each complex in v.2020 are shown in Table 2.1.

PDB entries	PDBbind complexes	Protein-ligand complexes	Protein-protein complexes	Protein-nucleic acid complexes	Nucleic acid-ligand complexes
157,974	23,496	19,443	2,852	1,052	149

Table 2.1: PDBbind v.2020 information [1]

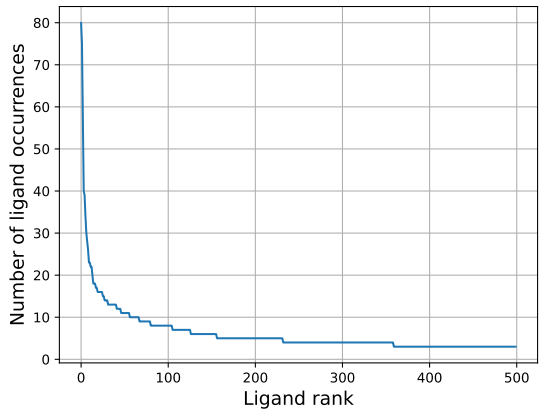
The authors of PDBbind create the dataset by searching PDB for valid complexes in the four major categories defined above. Out of 157,974 entries in PDB, 78,460 are valid complexes in v.2020. They then search the primary references for the experimental data in PDB to find the binding affinity data. This gives a general set consisting of 23,496 complexes. Furthermore, a refined set is provided which filters the general set down to 5,316 complexes of higher quality. The highest quality dataset is called the core set, which contained 285 complexes as of 2016, and is commonly used as a benchmark for docking and scoring problems, such as at the Comparative Assessment of Scoring Functions (CASF) [52].

Many machine learning papers that tackle docking or affinity prediction use PDBbind. Often the core set, CASF, is used as the test set due to its quality and repeated use as a benchmark. However, it is also common for these papers to use a timesplit to separate their training/validation/test data. By timesplit, we mean many papers will take the most recently deposited complexes to be thier test set, and will keep the older complexes for training/validation, some-

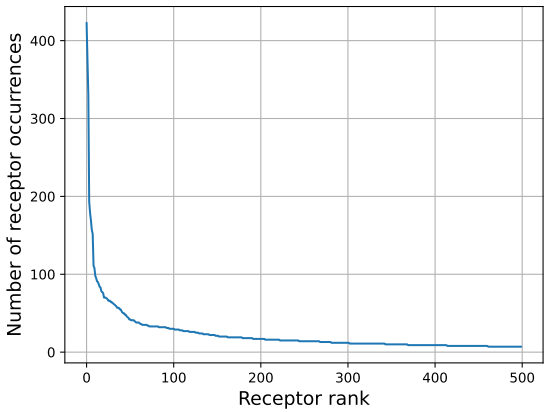
times using some filtering to avoid repeated ligands or receptors occurring in both the training and test sets.

We will now present some detailed information showing the characteristics of the PDBbind dataset (v.2020) general set of protein-ligand complexes. First of all, we applied a simple filter to the dataset in most of our experiments in this thesis. The majority of ligands in PDBbind are small ligands of mass around 200Da to 600Da. However, there are also some oligopeptides present, which are short peptide chains consisting of between 2 and 20 amino acids. These molecules are much larger than most drugs, and are much trickier to dock with classical docking methods due to their size, making them inappropriate for our docking experiments. Hence they were removed, filtering down the size of the dataset from 19,443 complexes to 16,844 complexes.

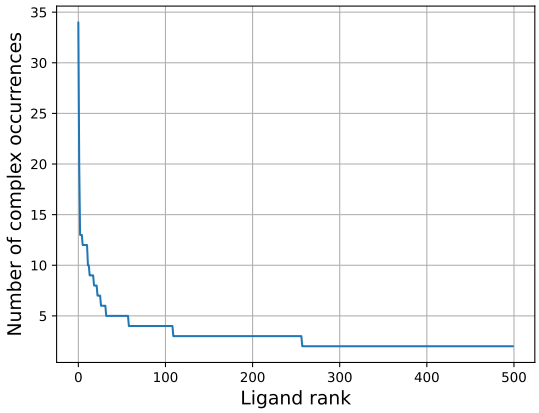
Next we calculate the number of unique ligands and receptors in the dataset. Out of 16,844 complexes, we calculate that there are 3,043 unique receptors, and 12,908 unique ligands. We did this by processing the index files that come with PDBbind to give the ligand and receptor names along with the complex name, which can then be filtered for duplicates. Other methods to find the number of unique molecules exist - for example, International Chemical Identifier (InChI) [20] keys are a standard way of encoding molecular information in a database which can be filtered for duplicates.



(a) Ligand frequency



(b) Receptor frequency



(c) Complex frequency

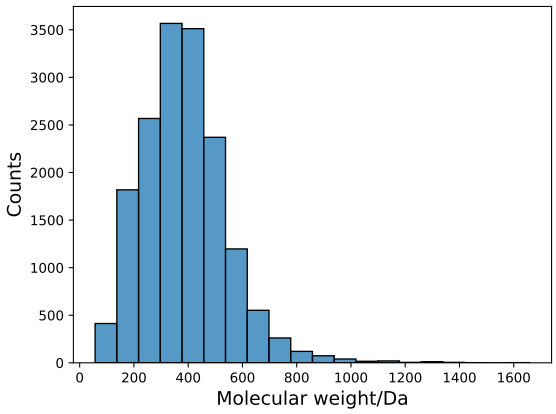
Figure 2.1: Histograms showing molecular frequencies in PDBbind

We additionally calculate that there are in fact only 15,277 unique complexes present in PDBbind. The duplicates are typically deposited from different experiments, and therefore

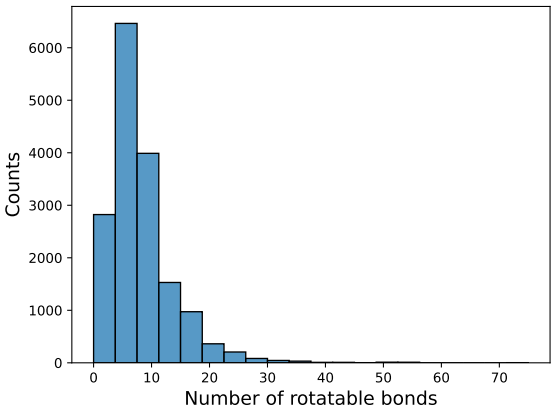
can have different resolutions and some experimental discrepancies, or they may be in different states (such as catalytic vs inactive enzyme states). This can lead to different methods of duplicate detection, such as InChI keys, having discrepancies with our results.

In Figure 2.1 we show histograms of the molecular frequencies for ligands, receptors, and complexes. These plots show the molecular frequencies ranked from most common to least common, cut off at 500 molecules for the purpose of visualisation.

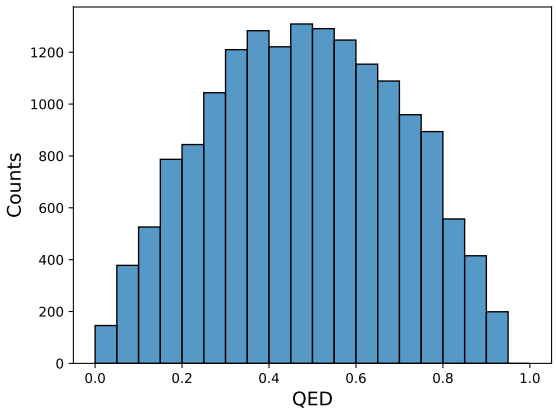
We will now briefly look at some metrics describing the ligands and the receptors, as these inform our data selection process in the redocking experiment in Section 4.1. For the ligands we show the distribution of molecular weights, of the number of rotatable bonds, the number of rings in each molecule, and the quantitative estimate of drug-likeness (QED). QED is a method of scoring how likely a molecule is to be a drug, with zero meaning very unfavourable and one being very favourable. It is computed by accounting for a range of molecular properties, improving upon the older Lipinski rule of five [31]. For detailed information see Bickerton et al. [6]. Plots showing the distributions of these four properties are shown in Figure 2.2, with information about these distributions given in Table 2.2.



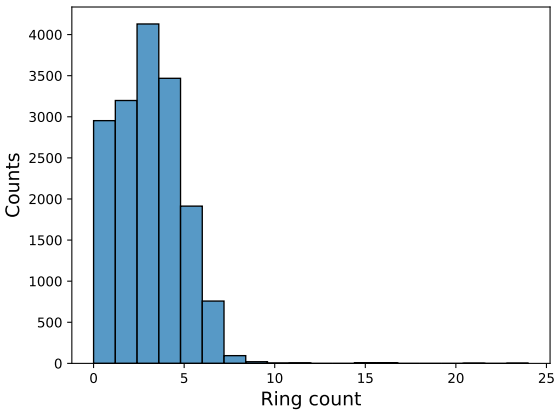
(a) Weight frequencies



(b) Rotatable bond frequencies



(c) QED frequencies



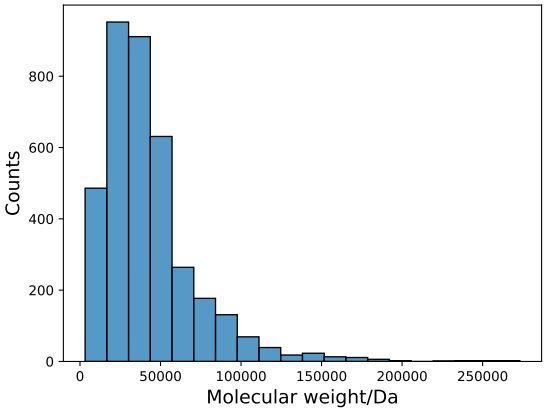
(d) Ring count frequencies

Figure 2.2: Molecular properties of PDBbind ligands

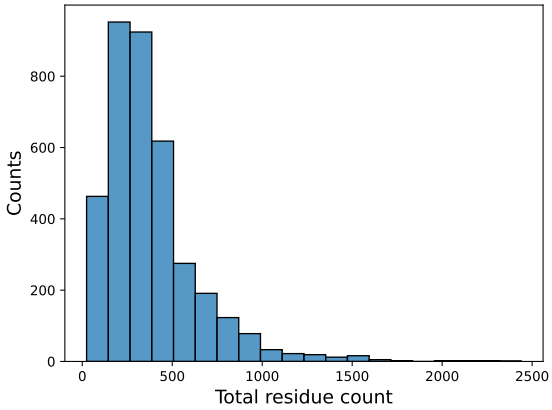
For the proteins we show three metrics - the molecular weights, the total residue count (number of amino acids counted across all protein subunits), and the number of subunits (individual chains) in each protein. Plots showing the distributions for these three metrics are shown in Figure 2.3, and information about these distributions is given in Table 2.3.

	Molecular weight/Da	QED	Number of rotatable bonds	Ring count
mean	386.03	0.482	8.05	3.03
std	156.80	0.214	5.71	1.64
min	56.88	0.014	0.00	0.00
25%	280.22	0.318	4.00	2.00
50%	375.38	0.483	7.00	3.00
75%	472.44	0.649	10.00	4.00
max	1659.41	0.947	75.00	24.00

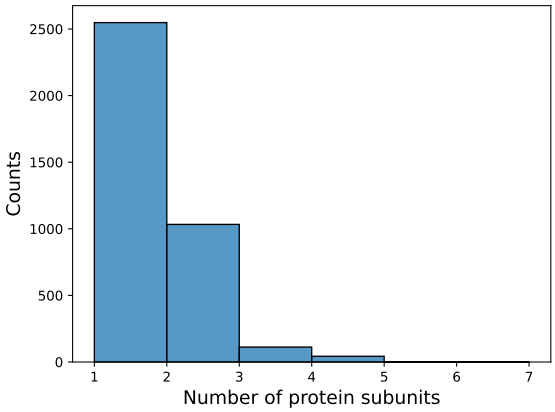
Table 2.2: Distribution of four metrics for the ligands in PDBbind



(a) Weight frequencies



(b) Frequencies of total residue count



(c) Frequencies of protein subunit counts

Figure 2.3: Molecular properties of PDBbind proteins

We finish this section with a brief discussion of the quality of the data present in PDBbind. With large datasets it is inevitable that some data will be corrupted or poorly configured. It is therefore common to see datapoints discarded in the literature if there is a processing error at some point in the pipeline. We found this to be a fairly serious issue for certain molecules in our redocking experiment in Section 4.1, and had to tailor our data selection to avoid these issues. For example, we found some molecules where the structure files incorrectly labelled bonds as aromatic when they were not, and molecules where the SMILES string did not match the structure files, causing experimental issues. This is exactly why the core set exists - it provides a smaller well curated dataset without these kinds of issues.

	Molecular weight/Da	Number of polypeptide chains	Total number of residues
mean	42695.67	1.38	381.78
std	29283.56	0.62	262.34
min	3177.54	1.00	24.00
25%	24060.44	1.00	219.00
50%	35284.45	1.00	312.50
75%	51199.64	2.00	459.25
max	273124.26	7.00	2440.00

Table 2.3: Distribution of three metrics for the proteins in PDBbind

2.2 File types

We will now briefly discuss the form of the structure files given in PDBbind. Proteins are all given as pdb files, which is a file type created for the PDB database, with extension .pdb. The primary purpose of pdb files is to describe the full quaternary structure of a protein - i.e., the 3d coordinates of each atom present in the protein. Additionally, they provide annotations including the authors and experimental procedures, primary structure (the amino acid sequences of the polypeptide chains) and secondary structure (whether each amino acid is in an alpha helix, beta sheet, beta turn, or omega loop). An example pdb file is shown in Listing 2.1.

Listing 2.1: Pdb file for protein 7mch

```
1 HEADER     LYASE           02-APR-21   7MCH
2 TITLE      CRYSTAL STRUCTURE OF A SINGLE-CHAIN E/F TYPE BILIN LYASE-ISOMERASE
3 TITLE     2 MPEQ IN SPACE GROUP C2221
4 COMPND     MOL_ID: 1;
5 COMPND    2 MOLECULE: BILIN LYASE-ISOMERASE;
6  ...
7 REMARK    2
8 REMARK    2 RESOLUTION.     2.95 ANGSTROMS.
9  ...
10 SEQRES    1 A  416  MSE GLY SER SER HIS HIS HIS HIS HIS HIS SER GLN ASP
11 SEQRES    2 A  416  PRO ASN SER SER SER MSE ALA GLU ARG PHE ASP ASN LEU
12 SEQRES    3 A  416  VAL GLU GLY LEU THR GLU GLU ARG ALA MSE ALA VAL ILE
13 SEQRES    4 A  416  LEU ALA ASP PRO ASP SER LEU GLU ARG PRO VAL ASP LYS
14  ...
15 HELIX     1  AA1 PRO A    -4  SER A     0  5                                    5
16 HELIX     2  AA2 THR A   13  ILE A    21  1                                    9
17  ...
18 SHEET     1  AA1 2 ASN A    7  VAL A    9  0
19 SHEET     2  AA1 2 ASN B    7  VAL B    9 -1  O  VAL B    9   N  ASN A    7
20  ...
21 ATOM       1  N   ASP A  -5      166.327   6.085  46.732  1.00 83.24    N
22 ATOM       2  CA  ASP A  -5      167.617   5.506  46.350  1.00 89.31    C
23 ATOM       3  C   ASP A  -5      168.026   5.805  44.901  1.00 79.12    C
24 ATOM       4  O   ASP A  -5      168.245   4.866  44.138  1.00 86.13    O
25 ATOM       5  CB  ASP A  -5      167.594   3.964  46.521  1.00 99.49    C
26  ...
27 HETATM   320  N   MSE A  36      165.826  26.077  65.017  1.00 66.88    N
28 HETATM   321  CA  MSE A  36      165.157  24.795  64.872  1.00 65.98    C
29 HETATM   322  C   MSE A  36      164.091  24.642  65.909  1.00 57.33    C
30  ...
31 END
```

The first word in each line gives the record type for that line. Note that this is not the full pdb file: we have only shown the components most relevant to our discussion. Additionally, there are typically thousands of atoms in a protein, and correspondingly, thousands of ATOM records. There can also be many different record types not seen here, typically corresponding to different types of annotations and metadata, including details about the authors and journal in which the data was published.

Record types such as HELIX and sheet give the secondary structure of the protein. The three record types worth noting in more detail are SEQRES, ATOM, and HETATM. SEQRES gives the primary structure: the first letter indicates which polypeptide chain is being described (A, B, C and so on), the number then gives the length of that chain, and it then gives the amino acid residues in order.

The ATOM record describes the 3d coordinates of each atom in the protein. First the entry gives us a number identifying that atom, followed by the atom type (for example, CA refers to the α-carbon), then the amino acid type it belongs to, then the polypeptide chain code, and then a number indicating the position of the amino acid in the chain. The next three floating point numbers give the x-, y- and z-coordinates of that atom in Angstroms. The next two numbers are the occupancy and the temperature, followed by the atomic symbol. The details of occupancy and temperature are not important for our discussion here.

Finally, HETATM refers to heteroatoms. These are atoms present in the protein structure that do not belong to a polypeptide chain. Typically they belong instead to a small molecule cofactor. Many protein structures have such cofactors present - for example, hemoglobin contains two of both of its α and β polypeptide chains, and several iron-containing heme groups, which are not polypeptide chains but are crucial to its quaternary structure. The HETATM record follows the same pattern as the ATOM record, except that the amino acid residue is replaced with a cofactor identifier.

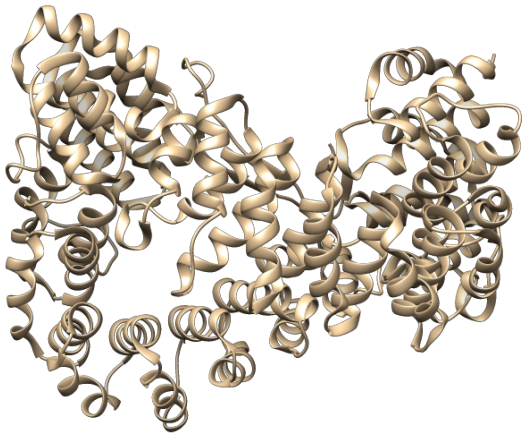


Figure 2.4: Visualisation of protein 7mch

A pdb file provides all the structural information necessary to study and process protein

structures. They can also be used to visualise a protein with a number of software packages. Throughout this thesis, we use *UCSF Chimera* [42] to generate images of proteins and ligands. An image of the protein described by the structure file above is shown in Figure 2.4. Note that the visualisation software gives an interactive 3D image of the protein, whereas we can only show a 2D image taken from one angle.

For more in depth information on pdb files we refer the reader to *The PDB Format Guide* [2].

The ligands in PDBbind are given as both sdf and mol2 files. These are both similar to pdb files in that they contain the three coordinates of each atom along with their atom types. They also contain explicit sections describing all the bonds present in the molecule along with the bond types. These files are optimised for smaller ligand-like molecules as opposed to the large proteins described by pdb files. We will not discuss the form of these files in detail as they are similar in essence to pdb files. For more information on both of these file types we refer the reader to *CTFile Formats* (PDF) by *MDL Information Systems* [53].

Simplified molecular-input line-entry system [58] (SMILES) strings are also provided for each ligand. SMILES strings specify the atoms present in a molecule and the topology (bond structure) in a one line string. They do not give any stereochemical information, i.e., no 3D structural information.

AutoDock Vina requires PDBQT files as input, which are slightly modified PDB files with a few additional features such as specifying bonds are rotatable [3].

2.3 Antibiotics

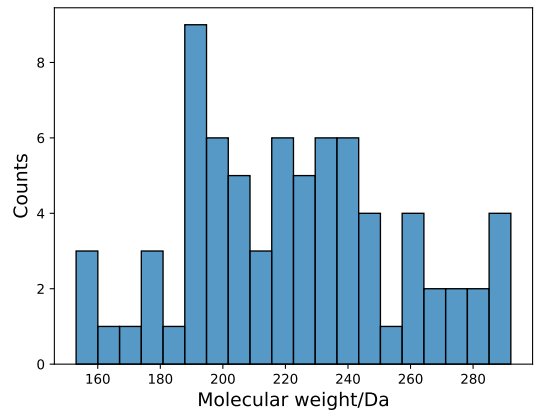
In Section 4.3 we run both classical and ML-based docking methods on a dataset generated by a mixed team from both April19 Discovery Inc. and UCL Computer Science. This dataset was generated as a part of the open source antibiotics challenge mentioned in Section 1.1. We now briefly outline the challenge, the pipeline used to generate the hits, and some metrics describing the hits.

Figure 2.5: Visualisation of MurD Ligase (PDB code 3uag)

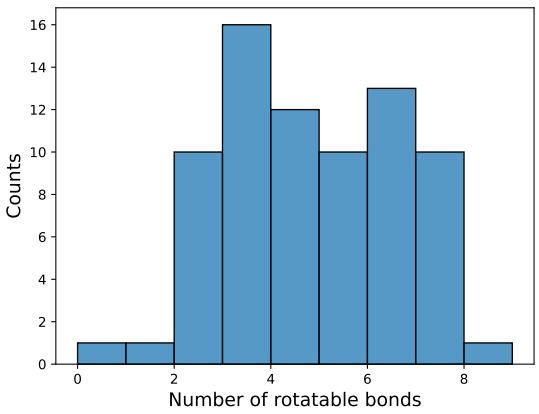
The target is the MurD Ligase, which has PDB code 3uag. We show a visualisation of this target in Figure 2.5. This is an enzyme present in many bacteria and crucial to their survival. The aim of the challenge is to find an inhibitor of this enzyme, which could work as an antibiotic.

The challenge follows the fragment-based lead discovery paradigm. Four small hits were identified experimentally. These hits do not bind in the active site of the enzyme, but in an alternative pocket. It is thought that this pocket may act as an allosteric site. Whereas the active site is the section of the enzyme directly involved in its catalytic action, an allosteric site is a different pocket to which small molecules can bind, leading to a conformational change in the protein, which may change its level of activity. The aim is to grow these fragments into hits that bind to this allosteric site, inhibiting the enzyme. For details about the fragments see: https://github.com/opensourceantibiotics/murligase/wiki/Initial-MurD-Hits.

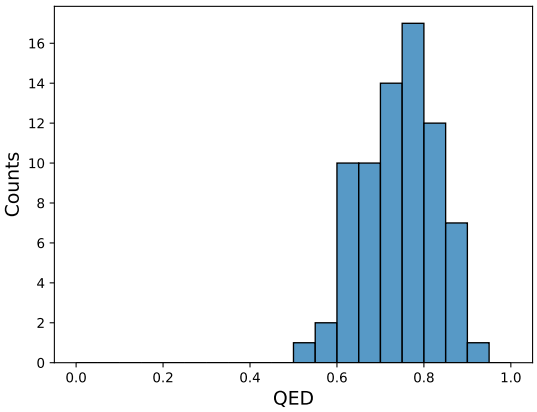
The April19/UCL then generated a dataset of 30 hits, with a pipeline including some cutting-edge ML models. The generative stage involved both a molecular variational autoencoder, and an evolutionary algorithm, to create potential hits. A number of filters were then applied to narrow these hits down, including some heuristic filters, a pharmacophore model, classical docking scores, and some visual inspection. The details of this pipeline are not directly relevant to the experiments done in this thesis, but are a nice proof of concept demonstrating the potential power of ML methods in the drug discovery pipeline. For more information about these methods see: https://github.com/opensourceantibiotics/murligase/issues/83.



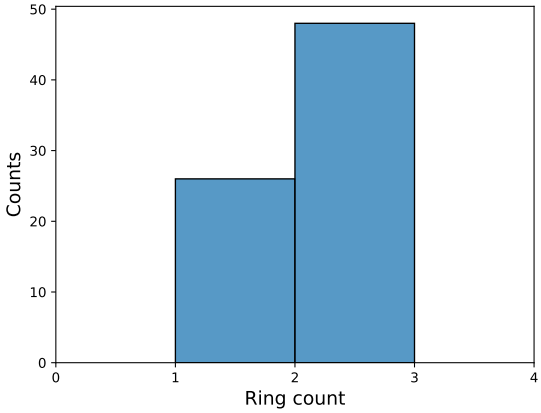
(a) Weight frequencies



(b) Rotatable bond frequencies



(c) QED frequencies



(d) Ring count frequencies

Figure 2.6: Molecular properties of the 75 generated hits in the antibiotics challenge

The dataset we use in Section 4.3 is comprised of a subset of 75 hits taken from this pipeline (which does not fully overlap with the final 30 hits). The SMILES strings for these 75 hits are shown in Appendix C. Histograms showing the distribution of four molecular properties - namely, the molecular weights, the number of rotatable bonds, the QED score, and the ring count, are shown in Figure 2.6.

2.4 DUD-E

The Directory of Useful Decoys - Enhanced [38] (DUD-E), is a database commonly used for benchmarking docking software. It provides a number of targets and active ligands, which are ligands that bind to the target. Additionally, for each active, it provides a number of decoys, which do not bind to the target.

Decoys are molecules with very similar global properties to the actives, but with different atoms, topology and stereochemistry. The properties matched between the decoys and actives in DUD-E are: the molecular weight, the estimated wateroctanol partition coefficient (miLogP), the number of rotatable bonds, the number of hydrogen bond acceptors, the number of hydrogen bond donors, and the net charge.

Decoys are designed to look similar to their associated active, but crucially, they do not bind to the receptor. They therefore provide a good test that a docking algorithm is actually using

structural information to dock the ligand and not just its global properties. While DUD-E was designed to benchmark classical docking software, we believe it may be even more relevant for ML-based docking software, as it is possible that an ML algorithm could learn to make predictions almost entirely based on the global properties of the ligands, if its training set is not very diverse.

DUD-E contains 102 targets. For each target, it has 100 to 600 active ligands, with an average of 224 actives per target. For each active it then has 50 generated decoys.

In particular, we make use of a subset of DUD-E called the diverse subset, which contains eight targets chosen to span a wide range of possible targets that docking software could encounter. We now list these eight targets with the DUD-E code in bold:

AKT1: Serine/threonine-protein kinase AKT

AMPC: Beta-lactamase

CP3A4: Cytochrome P450 3A4

CXCR4: C-X-C chemokine receptor type 4

GCR: Glucocorticoid receptor

HIVPR: Human immunodeficiency virus type 1 protease

HIVRT: Human immunodeficiency virus type 1 reverse transcriptase

KIF11: Kinesin-like protein 1

Chapter 3

Machine Learning Approaches

In this chapter we look at how machine learning (ML) techniques improve upon the classical methods discussed previously. In particular, we focus on the problems of docking, scoring, and hit detection, the latter of which is simply the prediction of whether a complex binds or not. We begin by discussing Morgan fingerprints, and then Extended Connectivity Interaction Features (ECIFs). We then give a brief outline of Geometric Deep Learning (GDL), then describe Protein-Ligand Interaction Graphs (PLIGs), and EquiBind.

3.1 Introduction

Binding affinity prediction, hit detection, and docking can be seen as supervised learning problems: we have some labelled experimental data and we want to learn from to predict the labels on unseen data. More formally, given a set of labelled datapoints $\{(x_1, y_1), ..., (x_N, y_N)\}$, the goal is to learn a function $f : X \to Y$ which makes predictions on unseen inputs x_j such that $y_j \approx f(x_j)$. This is often done by minimising some loss function $\frac{1}{N}\sum_i L(f(x_i), y_i)$ which characterises the closeness of predictions and labels. Additionally, we typically restrict f to some hypothesis space $f \in \mathcal{H}$ and add regularisation terms to the loss to increase the robustness of our model on unseen inputs and prevent overfitting.

For affinity prediction the label codomain Y is simply the set of real numbers $\mathbb{R}$, while for hit detection Y is a boolean, representing whether it binds or not. By contrast, the output for docking is a bound pose which lies in a highly complex space. Additionally, the inputs can be all sorts of complex objects: we have ligand SMILES strings, ligand structure files, and target structure files, all of which live in some very complex mathematical space.

Basic ML models, such as random forests or multi-layer perceptrons (MLPs), typically take a vector input, that is, $x \in \mathbb{R}^D$. In order to use such models, it is therefore necessary to map the complex inputs described above to a single real vector. Section 3.2 and Section 3.3 describe such mappings which can be used in conjunction with many ML models.

In contrast, more modern neural network architectures based on GDL can learn directly from the highly complex input and output spaces described above, without requiring some hand-crafted mapping to a feature vector. We meet two such methods in Sections 3.5 and 3.6.

We make another distinction between target-dependent and target-independent methods. Typically, basic fingerprint models are target dependent. That is, we map just the ligand to a vector, which is called a molecular fingerprint, and then train and test a ML model on the set of ligands with known binding data for a single target. Hence such models have to be trained

separately for each new target, and cannot work on targets with no experimental data. We discuss these methods in the section on molecular fingerprints.

By contrast, the remaining methods are trained on information about the target as well as the ligand, and can therefore be trained in a target independent way: that is, the same model can work across targets and hopefully even on unseen targets.

3.2 Molecular fingerprints

The most trivial approach to molecular fingerprinting would be to simply map global information about the ligand into a vector: for example, it could record the molecular weight, the ring count, the number of hydrogen donors etc. into a single vector. Such a fingerprint will certainly fail, since the topology (bond structure), and to some extent, the stereochemical information, is incredibly important for most prediction tasks. In particular, these fingerprints will make the same predictions for actives and decoys in DUD-E. We therefore move on to fingerprints which incorporate topological information.

Morgan fingerprints [35] were originally developed in 1965, not for machine learning, but as a way of storing chemical structural information in databases that can be easily queried, and used for purposes such as substructure querying and clustering. We first review not Morgan's original implementation, but an adaptation of this method which is optimised for predicting drug activity, called Extended-Connectivity FingerPrints [43] (ECFPs).

ECFPs aim to describe many of the substructures present in the molecule, such as rings and chains of different types. They are based on the topology of the molecule (the bond structure), but leave out 3D structural information. This is done through an iterative process, in which we begin with a list of atomic identifiers for each heavy atom present in the molecule. These identifiers are then updated to include information about their neighbours, several times, so that increasingly large substructures are described. Figure 3.1 shows an illustration of this process.

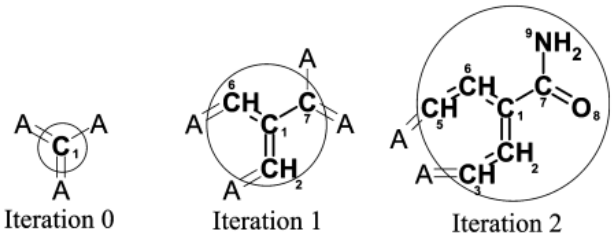


Figure 3.1: ECFP iterative updating (image credit Rogers and Hahn [43])

The first step is to assign a unique integer identifier to each heavy atom in the molecule. ECFPs do this using the Daylight atomic invariants rule: first they list for each heavy atom the number of heavy neighbours; the valence minus the number of hydrogens; the atomic number; the atomic mass; the atomic charge; and the number of attached hydrogens. They additionally add whether the atom is in a ring. These are put into a list, which is passed through a hash function which returns a single 32-bit integer.

For each update to the iterative process, a new list is created for each heavy atom, containing the iteration number, the atom's current identifier, and the bond types (1,2,3,4 for single, double, triple or aromatic bonds) along with the neighbours' current identifiers, sorted in a

deterministic order. This list is then hashed to create the new identifier for the atom. Note this process is performed for all heavy atoms in the molecule simultaneously.

The actual fingerprint begins with the list of atomic identifiers from iteration 0. At each iteration, new identifiers are appended to the fingerprint in a deterministic order. Note it is common for duplicates to appear due to the nature of the iterative procedure, so duplicates are removed. This procedure continues for a predetermined number of iterations, typically between two and six depending on the purpose.

The final fingerprint can then be put into a 2^{32} length one-hot-encoding, to create a fixed length fingerprint. This is an absurdly large vector, so it is typically then hashed into a 1024 bit vector. Bit collisions can then occur, since there are more than 1024 possible substructures, however, typically bit collisions are uncommon and do not present a problem. This final fingerprint describes many of the substructures present in the molecule.

This procedure is completely dependent on the hash function used, and even changing the version of the hash function will require models trained on these fingerprints to be retrained. However, any good hash function can be used.

A modification to ECFPs, called functional-class fingerprints (FCFPs), follow the exact same procedure, but use different initial atomic identifiers. ECFPs contain very specific atomic information, which may be undesirable for certain tasks such as binding affinity prediction, where very similar substructures may bind in the same way, and so we may want them to have the same descriptor. FCFPs therefore use more abstract initial atomic identifiers, composed of a six-bit one-hot encoding of the following features: if the atom is a hydrogen donor or acceptor; if it is negatively or positively ionizable; if it's aromatic; or a halogen. This is then hashed into the 32-bit initial identifier.

Note this has been a brief overview of ECFPs. For more detailed information, see Rogers and Hahn [43].

Many different ML models can be applied to these molecular fingerprints, including simple neural networks, support vector machines, and random forests. These all achieve similar performance on most targets for predicting hits [33]. For a comprehensive overview of the performance of these different ML models, see Chen et al. [10]. We do not restate their results here, but note that the performance of these models is very target-dependent, as they depend on the existing experimental data for any given target, which we see as problematic.

While ECFPs only depend on the topology of the ligand, an extension, known as the Extended 3D FingerPrint [4], also incorporates 3D structural information. It does this by updating each atom identifier with its neighbours defined not just by those bound to it, but also those within a certain spherical shell around the atom. The radius of this shell increases with each iteration. Axen et al. [4] test a few different related fingerprints, including also a 2D fingerprint which incorporates some stereochemical information. They find that for some tasks the structural information leads to a small performance boost.

The primary issue with the fingerprint methods discussed so far is that they only describe the ligand. Therefore, they need to be retrained for each new target. This makes them completely useless for new targets with no experimental data, although they can work quite well if you want to do virtual screening on a well studied target.

3.3 Extended Connectivity Interaction Features

This prompts us to move on to interaction features, which map a protein-ligand complex to a feature vector. Since these features contain information about the protein and its interaction, the hope is that models can be trained that are robust enough to work on unseen targets.

The catch is that these methods require a docked pose as input, since the interaction between the ligand and the target is characterised by their relative pose. These methods are therefore often used as a scoring function for docking. From the relative pose, we construct a feature vector (fingerprint), from which an ML algorithm can learn to predict a score for that pose. We can use this scoring function in conjunction with a search algorithm to construct a powerful docking program.

One such method is Extended Connectivity Interaction Features [46] (ECIFs). The idea behind ECIFs is very simple. We predefine a set number of unique ligand and protein atom types. In particular, atoms are defined by atom symbol, explicit valence, number of attached heavy atoms, number of attached hydrogens, aromaticity and ring membership. This gives a set of possible 'interactions' between all possible ligand types and protein types. An interaction is defined by the atom in the ligand and an atom in the protein being within a certain radius of each other. The number of interactions of each type is simply tallied, giving a feature vector with integer entries. The size of this vector can vary depending on the dataset - for example, in the dataset used by Sánchez-Cruz et al. [46], they had 70 unique ligand atom types and 22 unique protein atom types, giving a feature vector of length 1,540. The radius chosen can be varied to maximise performance on different tasks.

In the paper, as a proof of concept, they trained both a random forests model and a gradient-boosted trees model on their ECIFs to predict the binding affinity of a complex. They trained on the PDBbind refined set minus the core set, and tested on the core set. They did this experiment for a range of cutoff distances from 4Å to 15Å. Most of these models tested with a Pearson correlation coefficient of over 0.8, indicating strong performance. Interestingly, the models had a fairly low sensitivity to the cutoff radius, meaning fine-tuning this radius is not of great importance.

While these results are very impressive, we have one criticism of this paper. Since the length of their feature vector is defined by the dataset, their trained model won't run on a different dataset which includes different atom types without retraining. This can of course be accounted for by listing a larger set of possible atom types that don't actually occur in the training data. However, due to their encoding, such a model will not use unseen atomic interactions in a clever way, as it has no idea what that element of the feature vector corresponds to. Essentially, this model isn't truly learning the underlying physics of the problem. A more powerful model would be able to learn, for example, what happens when an atomic interaction it has not been trained on occurs, through similar atomic interactions in its training set.

3.4 Geometric Deep Learning

We now turn our attention towards models which can learn more directly from the input domain, without having to make use of a hand-crafted mapping to a feature vector. Just as deep learning improves on simple linear regression by learning the necessary features instead of them having to be predefined, geometric deep learning (GDL) can bypass the need for hand-crafted molecular features.

GDL is an emerging framework in which a wide variety of different deep learning architectures

can be understood collectively. We do not give a rigorous mathematical description of GDL, as this would diverge from the theme of the thesis. For this, we refer the reader to Bronstein et al. [9]. We instead give a brief conceptual overview of the subject and explain why we believe it paves the way forward for future docking and scoring models.

We return our attention to the inputs for a docking algorithm: we require a structure file for both the ligand and target. These are complex mathematical objects, unlike a simple feature vector. Two problems present themselves when trying to design a neural network that can learn from these objects. First, they are incredibly high dimensional, meaning we run immediately into the curse of dimensionality. Second, their dimensionality varies from input to input. For example, two different ligands can have different numbers of atoms, atomic coordinates, and bonds.

A classic analogy to the problem we face here comes from image processing. Say we have a grey-scale image that is 128 by 128 pixels. Early neural networks would simply flatten this into a 16,384 length feature vector. This is very high dimensional, so a neural network needs a huge training set to perform well.

The revolution in image processing came from convolutional neural networks [39] (CNNs). CNNs have translational invariance built into their architecture: that is, a small translation of the input does not change the model output. Building this invariance into the network actually involves removing a number of connections between each layer, and pinning other connections to the same value. It therefore massively reduces the number of weights that need to be learnt, side-stepping the curse of dimensionality. This makes intuitive sense: we know that an image of a duck is still an image of a duck with a small translation, so designing a network that has this built into its architecture removes the need for data augmentation - i.e., it doesn't need to learn from both the original and the slightly shifted duck. This dramatically reduces the size of the training set required to achieve good performance.

GDL generalises this idea. The underlying principle is to figure out the symmetry of the domain that your input data lies on, and then to build this symmetry into your network architecture. This can be conceptualised as a geometric prior: we know beforehand that our network should respect certain symmetries, so this is applied as a deterministic prior that any posterior model weights will respect.

Note there are two different ways we can respect the symmetry of the domain: through invariance, or equivariance. Invariance means the function output is unchanged by an input transformation, while equivariance means the output changes the same way that the input does. To illustrate this: imagine we want to classify an image as to whether it contains a duck or not, then a translation of the image should not change the classification and our function should be invariant. If instead we want the function to place a hat on the duck's head, then the hat should be translated with the duck, and so the function should be equivariant.

How exactly this is done depends on the input domain and its symmetries. Of particular relevance to us is two different input domains: graph-structured data, and Euclidean (spatial) data.

Graph neural networks [60](GNNs) work on graph-structured input data. We will give a very brief outline of how graph neural networks work but refer the reader to Bronstein et al. [9] for a more complete treatment. Graphs are composed of nodes and edges: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the edges connect different nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. If there are n nodes in the graph, the graph structure can be defined by an $n \times n$ adjacency matrix $\mathbf{A}$, where every element is binary and indicates whether that edge exists or not. We will only discuss undirected graphs in which case the adjacency matrix is symmetric. Additionally, we have a feature vector $\mathbf{x}_i \in \mathbb{R}^d$ at each node

$i \in \{1, ..., n\}$. These features can be appended into a $n \times d$ feature matrix $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_n)^T$. We also ignore graphs with edge features here. Hence a specific graph with data can be defined by the double $(\mathbf{X}, \mathbf{A})$.

Graph neural networks should respect permutation symmetry of the graph nodes, that is, it should not matter which way we order the nodes. A permutation matrix $\mathbf{P}$ reorders the nodes and edges of a graph as follows: $\mathbf{X} \to \mathbf{PX}$, $\mathbf{A} \to \mathbf{PAP}^T$. If we want to design a function that takes graph-structured data to a vector, it should be invariant to this transformation:

$$\mathbf{f}\left(\mathbf{PX}, \mathbf{PAP}^T\right) = \mathbf{f}\left(\mathbf{X}, \mathbf{A}\right) \tag{3.1}$$

Whereas if we want to design a function that transforms the node features of the graph to new features on the same graph structure, it should be equivariant to this transformation:

$$\mathbf{F}\left(\mathbf{PX}, \mathbf{PAP}^T\right) = \mathbf{PF}\left(\mathbf{X}, \mathbf{A}\right) \tag{3.2}$$

Where the function $\mathbf{F}$ gives the $n \times d$ feature matrix as output, and the adjacency matrix is unchanged.

The invariant function is very easy to implement: we just need to use a function that treats all the nodes exactly the same. For example, we could sum them or take the maximum of each element (maxpool).

The equivariant function is a bit trickier to implement. Bronstein et al. [9] group these into three families of increasing complexity, which we outline now.

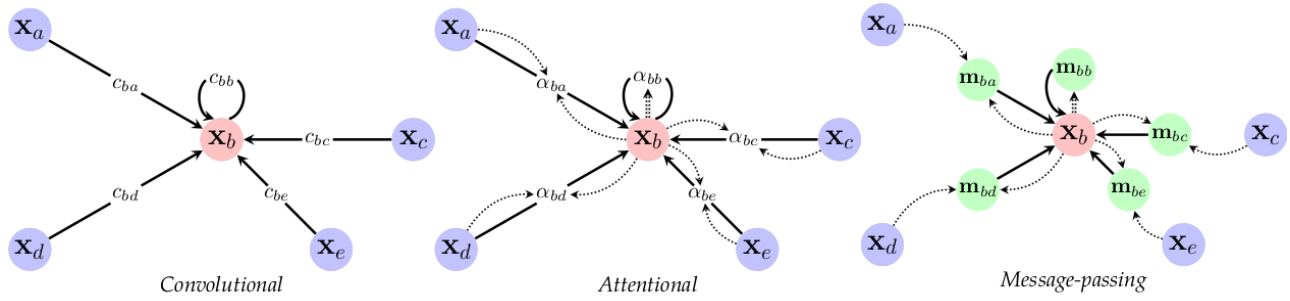


Figure 3.2: Different GNN families (image credit Bronstein et al. [9])

Throughout all the following layers, both ϕ and ψ are learnable functions (typically multi-layer perceptrons (MLPs)), while $\bigoplus$ represents a sum, mean, or maxpool. Graph convolutional networks apply the following update in each layer:

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right) \tag{3.3}$$

Here the c_{ij} are learnable parameters representing the importance of each neighbour in the updates.

Graph attention networks apply the following update:

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right) \tag{3.4}$$

The function $a(\mathbf{x}_i, \mathbf{x}_j)$ is an attention mechanism which means the weights are feature-dependent.

And finally, message-passing neural networks:

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right) \tag{3.5}$$

In which $\psi(\mathbf{x}_i, \mathbf{x}_j)$ is an explicitly learnt message sent along an edge as the function of both features, and is the most general GNN.

In many cases where we want to predict a non-graph output from graph-structured input data, our full GNN will consist of a few equivariant layers like those above, followed by an invariant layer like a maxpool to give a non-graph structured vector, finally followed by a simple MLP.

In addition to GNNs, we also encounter spatial data in $\mathbb{R}^3$. The symmetry that needs to be respected here is the Euclidean group of translations and rotations, SE(3), or E(3) if we include reflections. In fact, for our purposes, we encounter data on graphs that are also spatial. That is, each node has not only node features, but also node coordinates. For this task, we need to construct a neural network that respects both permutations of the node ordering, and translations/rotations of the node coordinates.

E(n) Equivariant Graph Neural Networks [47] provide a solution to this problem. We discuss just the E(3) case but this generalises to other dimensions. Formally, at each node, we have a feature vector $\mathbf{h}_i \in \mathbb{R}^d$, and a position vector $\mathbf{x}_i \in \mathbb{R}^3$. Each layer of this network is similar to the message-passing GNN above, but has different forms for the updates to the position vectors and feature vectors.

For the exact construction of the updates, see Satorras, Hoogeboom, and Welling [47]. Here we simply quote the result and you can check for yourself that the correct equivariance is respected.

$$\mathbf{m}_{ij} = \phi_e\left(\mathbf{h}_i^l, \mathbf{h}_j^l, ||\mathbf{x}_i^l - \mathbf{x}_j^l||\right) \tag{3.6}$$

$$\mathbf{x}_i^{l+1} = \mathbf{x}_i^l + C \bigoplus_{j \in \mathcal{N}_i} (\mathbf{x}_i^l - \mathbf{x}_j^l)\phi_x(\mathbf{m}_{ij}) \tag{3.7}$$

$$\mathbf{m}_i = \bigoplus_{j \in \mathcal{N}_i} \mathbf{m}_{ij} \tag{3.8}$$

$$\mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i) \tag{3.9}$$

As a final theoretical result, we briefly discuss Graph Matching Networks [29]. These neural networks take two graphs as input, and are trained to output a similarity score quantifying the similarity between the two graphs. We do not explicitly give the updates here, but note that they include messages passed within each graph and between each graph, in a way that respects the permutation equivariance of both graphs individually. For more information on these models, see Li et al. [29].

After this brief theoretical detour, we return to two models that use GDL architectures on molecular data.

3.5 Protein-Ligand Interaction Graphs

Protein-Ligand Interaction Graphs [34](PLIGs), like ECIFs, provide a description of a protein-ligand interaction, that can be trained with a neural network to provide a model that can predict binding affinity across a wide range of targets. The difference is that PLIGs return a graph-structured input for the neural network, rather than a hand-crafted mapping to feature vector, making this a potentially more powerful method.

The idea is to represent the ligand with a graph. Each edge represents a bond, and each node represents an atom, so this graph fully describes the topology of the ligand. Each node has five ligand atom features, namely: the number of adjacent heavy atoms; number of adjacent hydrogens; explicit valence; aromaticity; and ring membership.

In addition to this, 3D interaction information with the protein is incorporated though 22 protein interaction features at each node. These features are very simply: only 22 different atom types are found in the proteinogenic amino acids. Each column tallies the number of atoms of each atomic type that are within 4Å of the node.

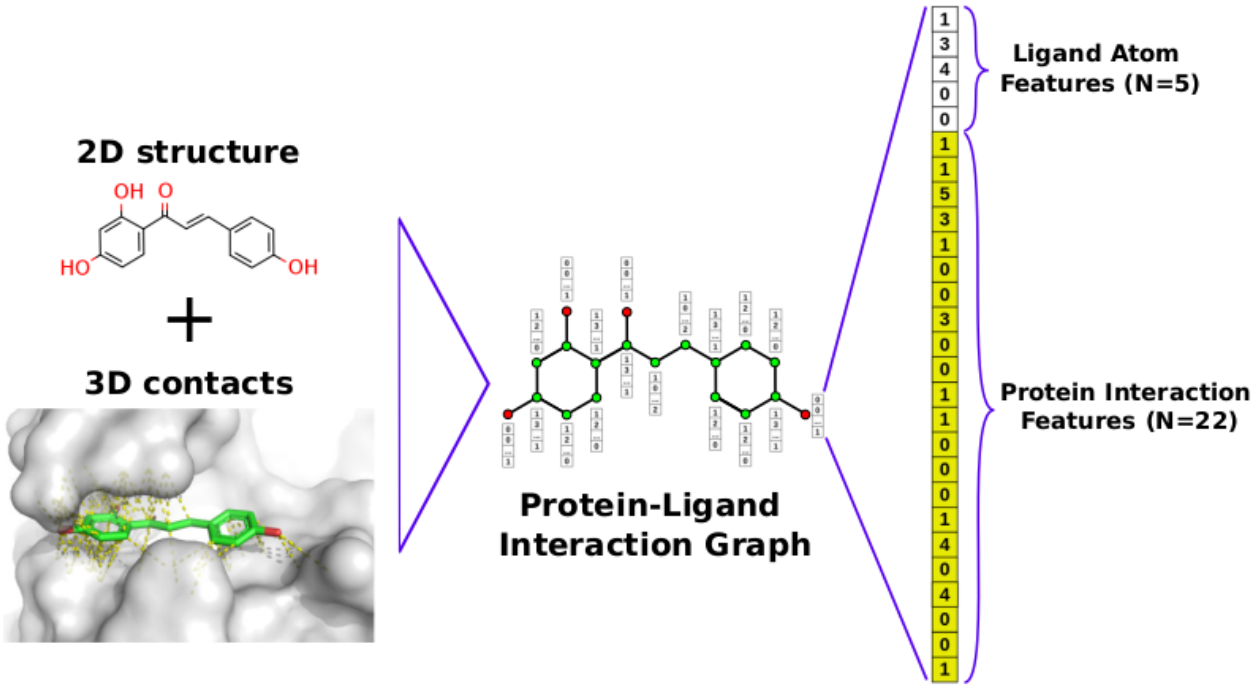


Figure 3.3: Protein-Ligand Interaction Graphs (image credit Moesser et al. [34])

The authors trained a number of GNNs on these graphs to predict binding affinity. Training was done on the PDBbind general set, and testing done on the PDBbind core set (CASF). Some of their models also supplemented the PLIG with an embedding of the protein sequence.

The best performing GNN they found was the Graph Attention Network, with ρ=0.84 (Pearson correlation coefficient), RMSE=1.22pK. This was slightly outperformed by a simple multilayer perceptron trained on ECIFs, with ρ=0.84, RMSE=1.19pK.

From these results, it appears that there is no benefit to using PLIGs compared to ECIFs for binding affinity prediction. However, we believe that PLIGs could be improved upon. In particular, the ligand atom features are very uninformative, they don't even include the atomic mass. Furthermore, the radius used for counting interactions was just set to 4Å, when this really should be a hyperparameter that is optimised. We believe that with some tweaking PLIGs should perform better than ECIFs, as they could contain all the same interaction information, while respecting the graph structure of the ligand.

3.6 EquiBind

EquiBind [50] is a GDL based docking model. It takes a ligand structure file and receptor structure file as input, and then returns a new ligand structure file giving the docked pose of the ligand relative to the receptor. It is able to model ligand flexibility, but keeps the receptor rigid. It respects E(3) symmetry as follows: its output is equivariant relative to the receptor position and orientation, and is invariant to the input ligand position and orientation. It is not invariant to the input ligand conformation - an effect that we explore in Chapter 4.

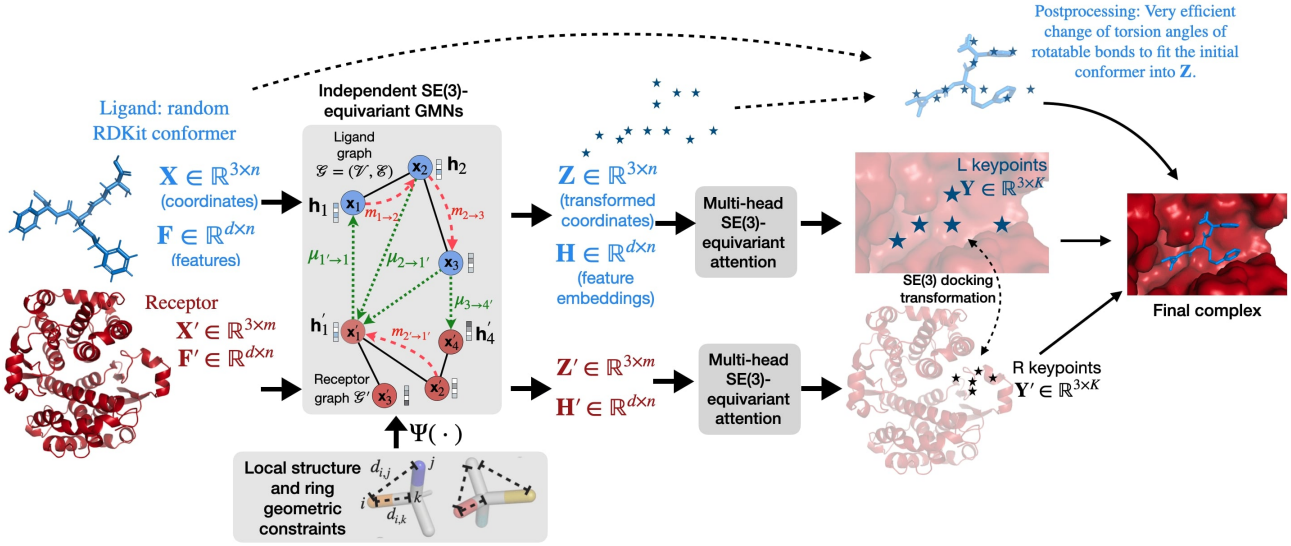


Figure 3.4: EquiBind architecture (image credit Stärk et al. [50])

The major innovation behind EquiBind is the creation of Independent SE(3)-Equivariant Graph Matching Networks (IEGMNs), first implemented by Ganea et al. [18] in their previous work EquiDock, which tackles rigid protein-protein docking. This network architecture builds on Graph Matching Networks [29] and E(n) Equivariant Graph Neural Networks [47] seen previously, to create a model that can perform graph matching between two graphs that have both features and spatial coordinates at every node, and respect the necessary symmetries.

This architecture is perfectly suited to the problems of protein-protein or protein-ligand docking. A graph with features and spatial coordinates is perhaps the most natural possible representation of a large molecule, as it captures the topology and the structure of the molecule. Furthermore, docking requires learning how to match the two molecules together based on their chemical and geometric features - the optimal docking pose typically requires structural similarity at the pocket, so the two molecules fit together well, and complementary chemical features such that they bind to each other strongly. Hence IEGMNs present the perfect architecture to learn to match the relevant chemical and structural features.

IEGMNs are based on message-passing GNNs. At each iteration, two different types of messages are passed: those within each graph, which pass between all neighbours, and those between the two graphs, which pass from every node to every node. The messages within the graphs are a function of the absolute displacement between neighbours, to ensure that the messages are SE(3) invariant. These messages then *update* the spatial coordinates of each node proportionally to the displacement between neighbouring nodes to ensure the spatial updates are SE(3) equivariant. The messages passed between the graphs are spatially independent linear functions of the graph features, that are multiplied by an attention mechanism that learns the importance of nodes relative to each other in opposite graphs as a function of their features. For example, it may learn under the hood what types of atoms bind to each other, and may therefore weight this interaction as being more important. Both the inter- and intra-graph messages are then fed through an MLP to produce the feature update for each node.

We now list all the updates in a single IEGMN layer used in EquiBind. As a word on notation, we denote graph 1: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with node features $\mathbf{F} \in \mathbb{R}^{d \times n}$, and node coordinates $\mathbf{X} \in \mathbb{R}^{3 \times n}$. Additionally, EquiBind makes use of edge features $\{\mathbf{f}_{i \to j} : \forall (i,j) \in \mathcal{E}\}$, which do not get updated but inform the node updates. The original (layer 1) node features are denoted $\mathbf{f}_i$ and also inform the updates. We denote graph 2: $\mathcal{G}'$, and all symbols that refer to it are the same but primed. $\mathbf{m}$ refers to messages passed within each graph, while μ refers to messages passed between the graphs. Each φ refers to a different small MLP. There are n atoms in the ligand and m atoms in the receptor.

$$\mathbf{m}_{j \to i} = \varphi^e(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, ||\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}||^2, \mathbf{f}_{j \to i}), \forall (i,j) \in \mathcal{E} \cup \mathcal{E}' \tag{3.10}$$

$$\mu_{j' \to i} = a_{j' \to i} \mathbf{W} \mathbf{h}_{j'}^{(l)}, \forall i \in \mathcal{V}, j' \in \mathcal{V}' \text{or } i \in \mathcal{V}', j' \in \mathcal{V} \tag{3.11}$$

$$a_{j' \to i} = \frac{\exp\left(\langle \varphi^q(\mathbf{h}_i^{(l)}), \varphi^k(\mathbf{h}_{j'}^{(l)}) \rangle\right)}{\sum_{k'} \exp\left(\langle \varphi^q(\mathbf{h}_i^{(l)}), \varphi^k(\mathbf{h}_{k'}^{(l)}) \rangle\right)} \tag{3.12}$$

$$\mathbf{m}_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{j \to i}, \forall i \in \mathcal{V} \cup \mathcal{V}' \tag{3.13}$$

$$\mu_i = \sum_{j' \in \mathcal{V}'} \mu_{j' \to i}, \forall i \in \mathcal{V}, \text{and } \mu_i' = \sum_{j \in \mathcal{V}} \mu_{j \to i'}, \forall i \in \mathcal{V}' \tag{3.14}$$

$$\mathbf{x}_i^{(l+1)} = \psi\left(\mathbf{x}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}}{||\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}||} \varphi^x(\mathbf{m}_{j \to i})\right) \tag{3.15}$$

$$\mathbf{h}_i^{(l+1)} = (1 - \beta)\mathbf{h}_i^{(l)} + \beta \varphi^h(\mathbf{h}_i^{(l)}, \mathbf{m}_i, \mu_i, \mathbf{f}_i), \forall i \in \mathcal{V} \cup \mathcal{V}' \tag{3.16}$$

After several IEGMN layers, the output coordinates of both the ligand and receptor are individually transformed into 'keypoints' which represent the docked pose, via an SE(3)-equivariant attention mechanism, where the attention weights are informed by the node features of the ligand or receptor graph as follows:

$$\mathbf{y}_k = \sum_{i=1}^{n} \alpha_i^k \mathbf{z}_i \tag{3.17}$$

$$\mathbf{y}'_k = \sum_{i=1}^{m} \beta_j^k \mathbf{z}'_j \tag{3.18}$$

$$\alpha_i^k = softmax_i \left(\frac{1}{\sqrt{d}} \mathbf{h}_i^T \mathbf{U}_k \mu(\varphi(\mathbf{H}')) \right) \tag{3.19}$$

And the β attention coefficients are defined similarly, with the graphs switched over. μ represents the mean operation, and $\mathbf{U}_k$ is a learnable matrix for each attention head.

The purpose of the keypoints is to represent the docked ligand pose. Both the receptor and the ligand produce keypoints, and we note that the keypoints are equivariant for both the ligand and the receptor (they are not aligned). The keypoints need to be aligned as best as possible to give the final docked pose. This is done with the Kabsch algorithm [25], which is a well-established algorithm which gives the rotation and translation that minimises the RMSD between two point clouds in 3D space. This is used to output the rotation and translation that best maps the ligand keypoints to the receptor keypoints, and it is this exact same rotation and translation that is applied to the input ligand to give the output mode.

EquiBind can be used in a mode where the ligand is rigid, in which case, this rotation and translation are all that is needed. However, it can also model ligand flexibility. It models ligand flexibility just through rotatable bonds, and finds the output conformation by rotating the bonds of the input ligand conformation to best align with the output position coordinates $\mathbf{Z}$ of the final IEGMN layer. This is not a trivial problem as many ligand conformations are implausible, due to, for example, the ligand overlapping with itself. Additionally, solutions such as gradient descent in the torsion angles can be very expensive to compute. The authors present a novel 'fast point cloud ligand fitting' algorithm that performs this optimisation with no optimisation required - i.e., it has a closed form solution and is very fast to compute. In short, the authors describe the space of possible conformations with a probability distribution given by the von Mises distribution (circular normal distribution) for each rotatable bond, centred on the dihedral angles of $\mathbf{Z}$. The output conformation is then the maximum likelihood estimate of this probability distribution, subject to the constraints, which has a closed form.

We now briefly discuss the featurisation of the protein and ligand. The protein is represented by a graph where each node represents an amino acid, centred on the α-carbon. The node feature is a one hot encoding of the residue type. The edge features represent both the distance between residues, and the angle between them, given by a locally defined coordinate system. For details, see Ganea et al. [18]. The ligand graph has a node for every atom, with node features: atomic number; chirality; degree; formal charge; implicit valence; the number of connected hydrogens; the number of radical electrons; hybridization type; whether or not it is in an aromatic ring; in how many rings it is; and 6 features for whether or not it is in a ring of size 3, 4, 5, 6, 7, or 8. The ligand has the same edge features as the protein.

The loss function used in EquiBind most notably depends on the centroid displacement, the ligand RMSD, and the ligand Kabsch RMSD (which is the minimum possible RMSD following an optimal rotation and translation). This is taken between the predicted pose and the experimental pose.

EquiBind is trained and tested on PDBbind. Instead of using the core set (CASF), the authors use a timesplit, randomly sampling 125 proteins from a set of 1,512 new complexes

deposited in PDBbind since 2019, and collecting the new complexes with these proteins to create the test set. They leave the rest for training and validation, but remove complexes such that no ligands appear in both the training/validation and the test set. Some receptors appear in both sets, but some receptors in the test set are unseen.

EquiBind's performance is very impressive: in terms of the metrics centroid displacement, RMSD, and Kabsch RMSD, it often slightly outperforms, or at least matches, a number of classical docking software. In addition, it has an immense computational speedup: EquiBind made predictions at an average of 0.16s per complex on a 16-core CPU, compared to 49s for *QVINA-W*, 146s for *SMINA*, 247s for *GNINA*, and 1,405s for *GLIDE*.

This is a truly incredible achievement, as this computational speedup makes it possible to screen far larger libraries for virtual screening. We believe the future of virtual screening will follow the blueprint of EquiBind, and will use sophisticated neural networks based on GDL principles.

However, this model is in its infancy. It has only been tested on a small test set, composed of crystal structures. This is in comparison to classical docking software, for which the major methods have been tested and analysed in dozens of research papers. It has not been tested on decoys. Additionally, there is the issue of protein flexibility. In this case, we believe PDBbind may make the task easier than it is realistic to expect in a commercial setting. When a ligand binds to a protein, the protein changes its conformation slightly to accomodate the ligand. If we measure this bound experimental structure, then remove the ligand from the protein, the protein's pocket may be slightly conformed to fit the ligand. This makes the prediction task potentially much easier than predicting the same docked structure from the unbound experimental structure, or from a predicted protein structure using software such as AlphaFold [24].

One of the primary aims of the experimental work done is this chapter is to begin this rigourous assessment of the behaviour of EquiBind, to highlight areas for improvement so that it can be used successfully for drug discovery.

Chapter 4

Experiments

The overarching aim of the experimental work done in this chapter is to rigorously assess the behaviour of EquiBind in a wide variety of tasks. In particular, we want to investigate our two hypotheses, which we restate now:

Hypothesis 1: EquiBind has comparable performance to classical docking methods on targets in its training set, but is not a useful docking method on unseen targets.

Hypothesis 2: EquiBind learns implicit 3D fingerprints in its layers that can be generalised to other prediction tasks, including binding affinity prediction.

To evaluate our first hypothesis we perform two different redocking experiments, in which we dock ligands using AutoDock Vina, and then redock these poses using EquiBind, on a diverse selection of targets from the training set and then the test set. First of all we redock the crystal structures of the ligands. However, this test doesn't represent how molecules are docked in virtual screening, in which we don't know the output conformation, so we additionally generate conformers which are then docked and redocked. Additionally, we then analyse the performance of EquiBind on the MurD Ligase from the antibiotics challenge, to demonstrate the challenges of using EquiBind in a production setting.

For our second hypothesis, we present EquiBind-score, in which we use different layers from EquiBind as 3D fingerprints, and train a graph neural network to predict binding affinities from these fingerprints. We conclude by analysing the performance of EquiBind-score on a set of decoys.

4.1 Redocking Crystal Poses

In this experiment we investigate the behaviour of EquiBind by docking experimental crystal poses with AutoDock Vina, then redocking these poses with EquiBind. We then measure the displacements of these poses relative to their ground truth pose. We do this for a number of ligands for each chosen target, and for a number of conformations predicted by AutoDock Vina for each ligand. While it is expected that any docking software will shift ligands to some degree, a target-wide bias for shifts in a certain direction indicates that the docking software is failing to find the pocket correctly, while a high standard deviation indicates a large degree of sensitivity to the input conformation.

Our hypothesis is that: *Both EquiBind and AutoDock Vina distribute poses evenly around the ground truth pose for targets in the training set, but EquiBind may shift poses away from the ground truth for targets in the test set, as it has not learnt the pocket location.*

To clarify any confusion about EquiBind's invariance properties: as discussed in Section 3.6, EquiBind is SE(3) invariant, which means that its output will be identical for any translation or rotation of the input ligand. However, EquiBind is not invariant to the ligand conformation. Hence we expect different results for different poses predicted by AutoDock Vina as they have different conformations. This experiment therefore tests the sensitivity of EquiBind to different input conformations.

4.1.1 Experiment design

Since classical docking methods like AutoDock Vina are computationally demanding and slow, and since we only have limited computational resources, we choose a subset of PDBbind to perform this experiment on.

First of all, we split the dataset into complexes in the EquiBind training or validation set, and those in the test set. Recall from Section 3.6 that Stärk et al. [50] performed their split by target, so no target is in both sets. We grouped the complexes by target, and then removed all targets which are in fewer than ten complexes.

For each complex we then calculated a number of metrics for the ligand: the molecular weight, the QED score, the NHOH count, the NO count, the number of rotatable bonds, and the ring count. Additionally, we calculated two metrics assessing the performance of EquiBind for each complex. The first is the RMS error, which takes the magnitude of the displacement of each atom between its EquiBind-predicted pose and experimental pose, sums their squares and roots the result. The other metric we refer to as the ECIF Frobenius norm. We first calculated the extended connectivity interactive features [46] (ECIF) for all experimental and EquiBind-predicted poses. ECIFs provide protein-ligand atom-type pair counts that provide a good descriptor of the interactions between the protein and the ligand. Each ECIF is a matrix. For each complex, we then took the difference between the experimental ECIF and the EquiBind-predicted ECIF, and then took the Frobenius norm [15] of this resulting matrix, which provides another metric describing the difference between the experimental and predicted poses.

Target	Ligand count	Set	Status	PDBbind code
Calmodulin-domain protein kinase 1	19	Train	Success	2wei
PA-I galactophilic lectin	11	Train	Success	4ywa
5'-AMP-activated protein kinase catalytic subunit A	11	Train	Success	2y8l
Tryptophan synthase alpha chain	16	Train	Success	2cle
tRNA (guanine-N1-)-methyltransferase	33	Train	Success	4mcd
3-phosphokimate 1-carboxyvinyltransferase	14	Train	Fail	1g6s
Arginase-1	19	Train	Fail	2aeb
GTPase KRas	18	Test	Success	6quw
ABC transporter, periplasmic substrate-binding protein	13	Test	Success	6jb4

Table 4.1: Targets chosen for redocking

We then calculated the mean and standard deviation of all these metrics across all ligands for each target. We used these metrics to pick a set of targets by hand that we feel broadly covers the range of targets present in PDBbind. In particular, we place emphasis on the performance of

each target, measured by the RMS error and ECIF Frobenius norm, and made sure we included targets with good performance, with bad performance, and with a range of performances (as measured by the standard deviation of these metrics). In Table 4.1 we present the targets chosen for this, along with the number of ligands, whether they are in the train or test set, and whether they were included in our final results or not, since we encountered processing issues for some targets. We detail these processing issues in Section 4.1.2. Additionally, we include the PDBbind code for the target. These targets can be visualised in Appendix A.

The structure of each complex in PDBbind is experimentally measured. As discussed in Section 1.3.1, receptors are flexible, and hence the target structures will be slightly different for the same target bound to different ligands. In order to make fair comparisons, we want to dock all the different ligands to exactly the same structure. Hence we choose the structure in PDBbind which has the best resolution from the experimental x-ray crystallography, ignoring any structures that were measured using NMR spectroscopy. This is the structure to which the PDBbind code refers in Table 4.1.

Each ligand was then docked to its target using AutoDock Vina. The ligands chosen were all MOL2 files representing the experimentally determine structures from the PDBbind database. AutoDock Vina requires a pocket to be specified for docking. For each chosen crystal pose, we calculated the mean coordinates of all the heavy atoms in the ligand which belongs to that complex, then specified a cubic box with width 25Å centred on this position as our pocket. AutoDock Vina requires that both the ligand and target are specified as PDBQT files. The target and ligand were both converted using the following AutoDock Vina comands. The options mean that hydrogens are explicitly added to the target, and then for both molecules, non-polar hydrogens are removed, charges are merged, and lone pairs are removed. These options additionally specify that the receptor is held rigid.

```
prepare_receptor4.py -r target.pdb -o target.pdbqt -A hydrogens -U nphs_lps -v
prepare_ligand4.py -l ligand.mol2 -o ligand.pdbqt -U nphs_lps -v
```

These molecules were then docked using AutoDock Vina with the standard settings. In particular, we used the default exhaustiveness setting (exhaustiveness=8). The configuration file simply specifies the pocket, ligand and target as described above.

```
vina --config config_dock --out output.pdbqt --log log.txt
```

When a target/ligand pair is docked successfully, AutoDock Vina returns nine poses, which are the nine lowest energy poses that it sampled. These poses are returned as a single PDBQT file. We split this file into nine individual PDB files using the following Open Babel command:

```
obabel -ipdbqt output.pdbqt -opdb -O ligand_.pdb -m
```

The ligand PDB files were then docked to the same target structure using EquiBind. EquiBind performs blind docking, so it is not necessary to specify the pocket. We used the 'flexible_self_docking' mode, which models ligand flexibility, but keeps the receptor rigid. We set 'use_rdkit_coords' to false, which means it uses the conformation specified by the PDB file as input, instead of generating a new conformer using RDKit.

Next we computed the centroid of each pose. We define the centroid to be the mean coordinate of all heavy atoms in the ligand. Note that this is different to the centre of mass as we do not account for atomic weight. The centre of mass is typically very skewed towards the heavier atoms in the ligand so we feel the centroid is a more appropriate location to use. We then subtract the centroid coordinates of the ground truth pose from the centroids of each pose, for both the outputs of the first dock with AutoDock Vina, and the outputs of the redocking with EquiBind. We define the ground truth pose to be the lowest energy pose predicted by

AutoDock Vina, as we are unable to use the experimental poses as this would give inconsistent results due to the issue of target flexibility discussed previously. Finally, we calculated the mean and standard deviation of these displacements in the x-, y- and z-directions across all ligands for each target, which we present as our results for analysis.

4.1.2 Results

For both *3-phosphokimate 1-carboxyvinyltransferase* and *Arginase-1* we encountered issues while docking with AutoDock Vina. Normally, nine low energy poses should be returned, but for a number of ligands fewer than nine poses were returned. When we inspected these poses, we found that a number of them were corrupted. For example, for some of them the molecules were split into two disjoint parts. While some ligands worked this issue was prevalent enough that we didn't have much data so we discard these two targets entirely.

In this section we present our results for two cherry-picked targets. The results for the other targets can be found in Appendix B, which we encourage the reader to look at as they also form the basis of our discussion.

Training set

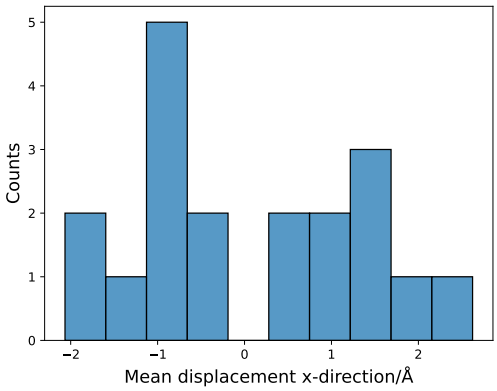
Calmodulin-domain protein kinase 1

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.087	0.114	0.089	1.575	1.687	1.859
std	1.316	1.548	1.708	0.540	0.816	0.804
min	-2.065	-1.637	-3.180	0.571	0.435	0.644
25%	-0.794	-0.990	-0.489	1.289	1.070	1.309
50%	-0.428	-0.261	0.493	1.478	1.553	1.667
75%	1.219	0.951	1.247	1.810	1.865	2.160
max	2.623	3.284	2.796	2.883	3.485	3.328

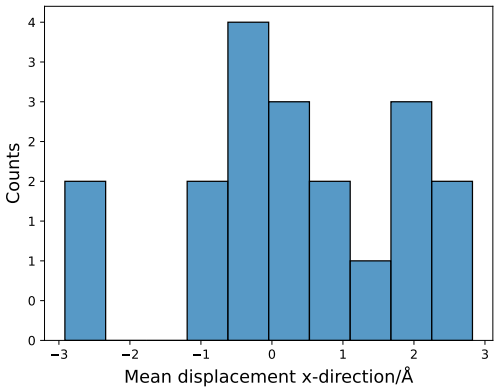
Table 4.2: Calmodulin-domain protein kinase 1 docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.321	-0.886	0.164	1.207	1.731	1.224
std	1.591	1.800	1.758	0.924	0.707	1.110
min	-2.916	-2.860	-3.249	0.078	0.389	0.105
25%	-0.428	-2.224	-0.485	0.355	1.281	0.396
50%	-0.015	-1.414	0.341	0.915	1.630	0.821
75%	1.676	0.161	0.856	1.945	2.443	1.747
max	2.824	2.700	3.588	2.750	2.702	3.385

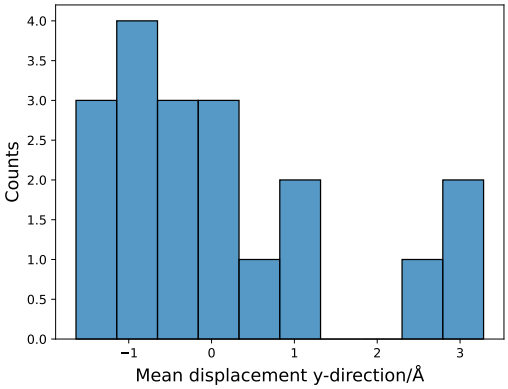
Table 4.3: Calmodulin-domain protein kinase 1 redocked metrics/Å



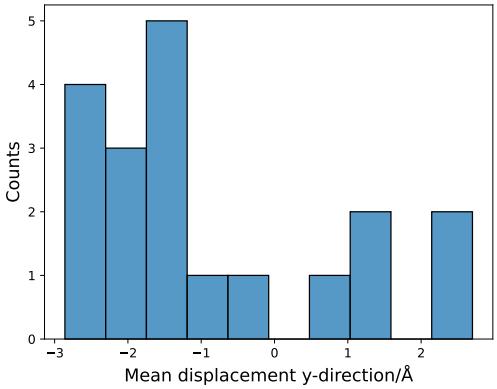
(a) Docked



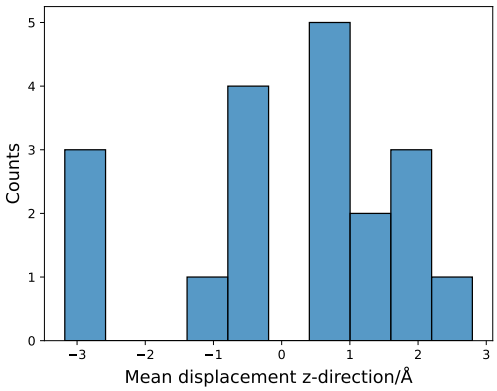
(b) Redocked



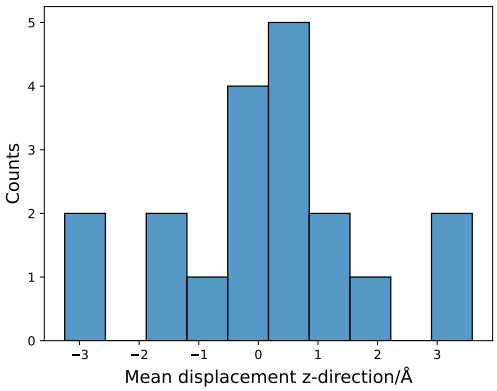
(c) Docked



(d) Redocked

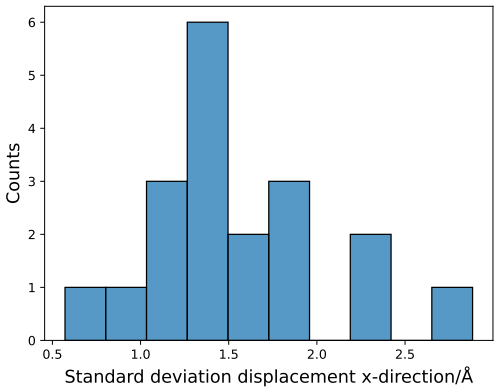


(e) Docked

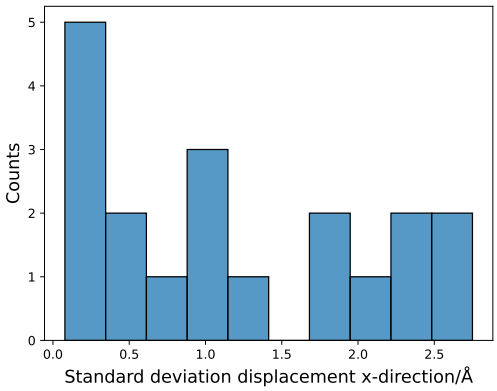


(f) Redocked

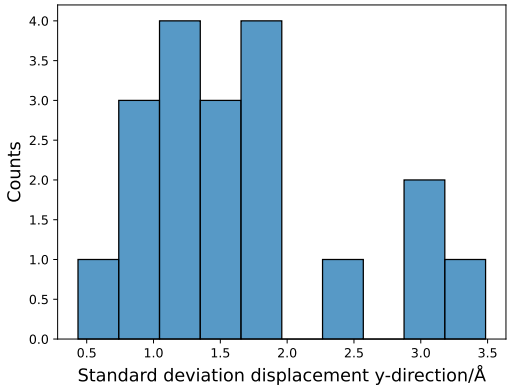
Figure 4.1: Calmodulin-domain protein kinase 1 mean displacements



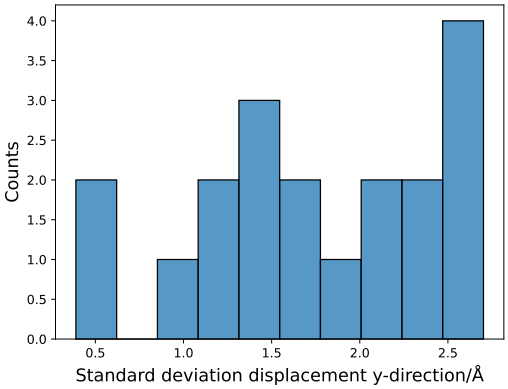
(a) Docked



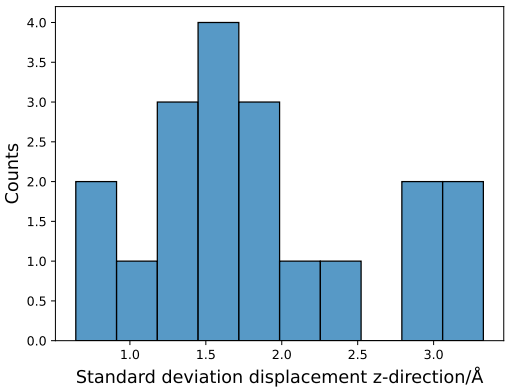
(b) Redocked



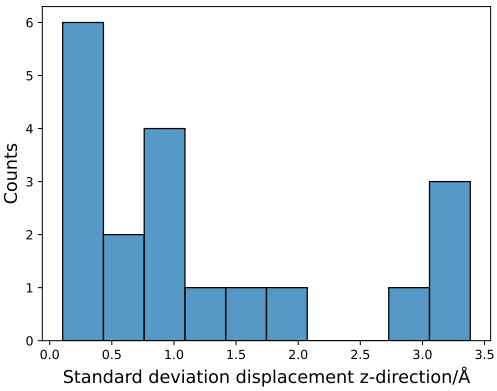
(c) Docked



(d) Redocked



(e) Docked

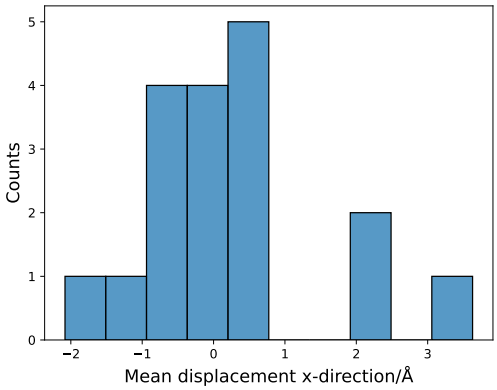


(f) Redocked

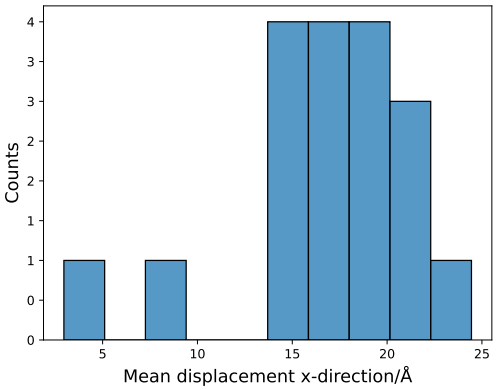
Figure 4.2: Calmodulin-domain protein kinase 1 standard deviation displacements

Test set

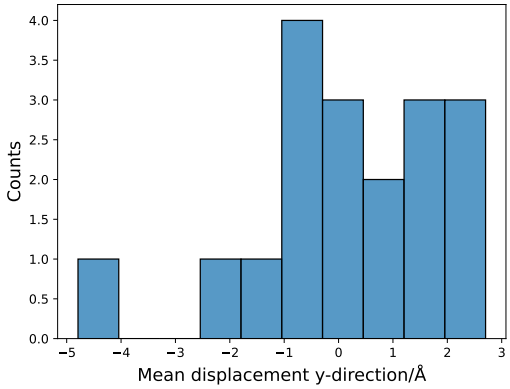
GTPase KRas



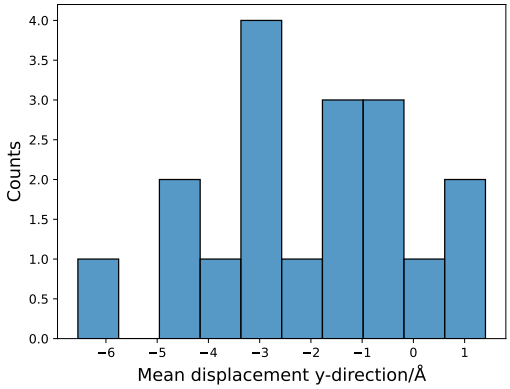
(a) Docked



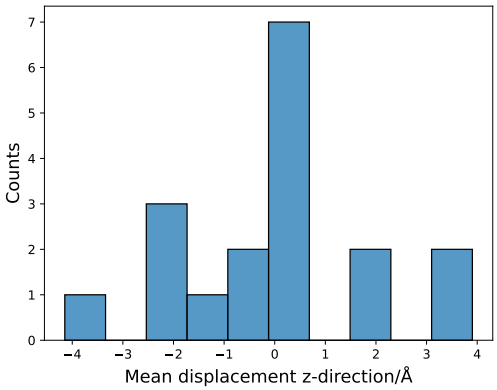
(b) Redocked



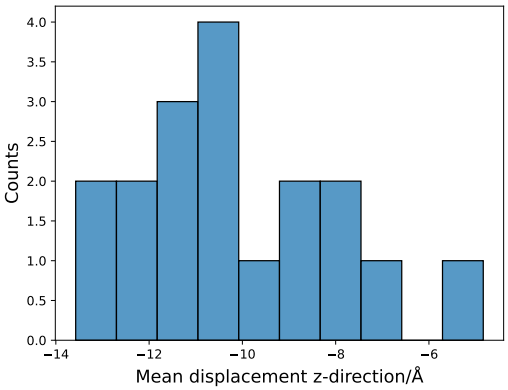
(c) Docked



(d) Redocked

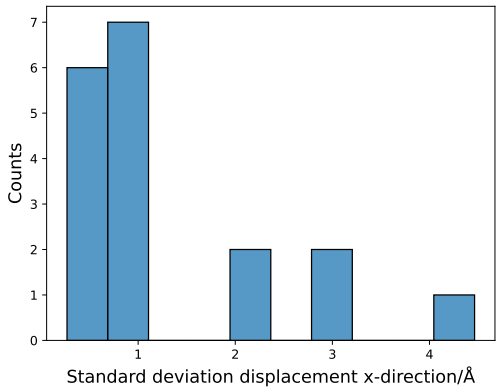


(e) Docked

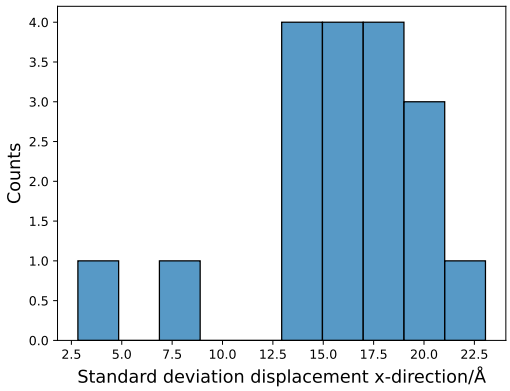


(f) Redocked

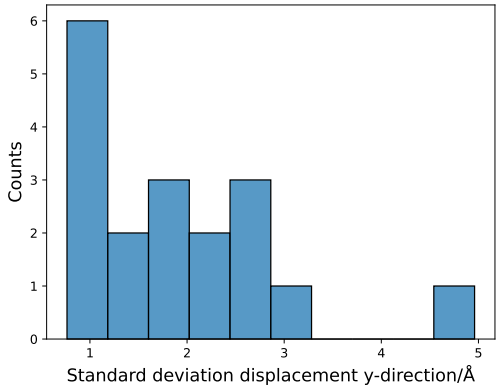
Figure 4.3: GTPase KRas mean displacements



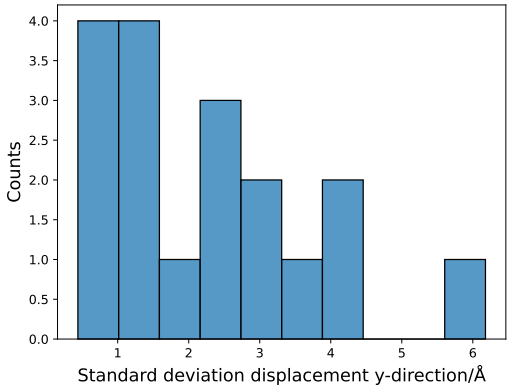
(a) Docked



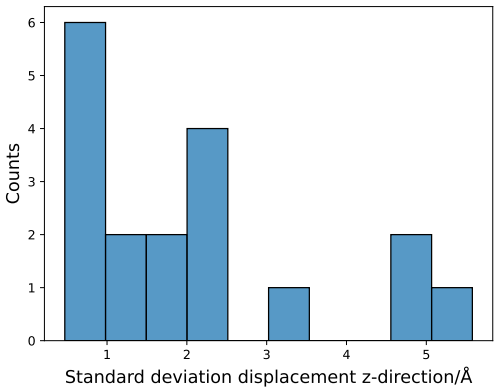
(b) Redocked



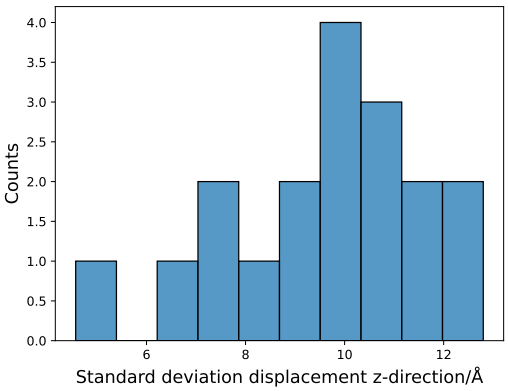
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure 4.4: GTPase KRas standard deviation displacements

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.235	0.246	0.024	1.339	1.905	2.103
std	1.305	1.851	1.943	1.157	1.051	1.576
min	-2.082	-4.792	-4.147	0.267	0.764	0.471
25%	-0.508	-0.559	-1.114	0.633	1.110	0.930
50%	0.075	0.089	-0.083	0.950	1.763	1.768
75%	0.429	1.689	0.639	1.851	2.400	2.327
max	3.629	2.703	3.905	4.464	4.960	5.577

Table 4.4: GTPase KRas docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	16.509	-1.995	-10.133	15.581	2.267	9.568
std	4.918	2.108	2.154	4.625	1.553	2.023
min	2.958	-6.547	-13.574	2.823	0.440	4.570
25%	14.826	-3.231	-11.332	13.999	1.079	8.661
50%	16.631	-2.127	-10.406	15.690	2.031	9.815
75%	19.442	-0.407	-9.144	18.336	3.093	10.687
max	24.448	1.406	-4.835	23.050	6.178	12.800

Table 4.5: GTPase KRas redocked metrics/Å

4.1.3 Discussion

We discuss each target in turn. *Calmodulin-domain protein kinase 1* performs as well as AutoDock Vina, as expected, indicating that EquiBind can perform very well on easy targets in the training set.

PA-I galactophilic lectin performs poorly, systematically shifting ligands about 10Å from the ground truth pose. We believe this is due to the structure of this protein, which is a homodimer, meaning it is comprised of two identical subunits bound together to form the quaternary structure. Since EquiBind performs blind docking, it has no preference for either of the identical pockets in either subunit. This therefore gives it a very large mean displacement and standard deviation. While this behaviour is interesting in its own right, it means a comparison with AutoDock Vina here is unfair

5'-AMP-activated protein kinase catalytic subunit A performed very poorly despite being in the training set. As indicated by its name, this target is just one subunit of the full protein *5' AMP-activated protein kinase*. This protein is composed of three subunits, and is activated by adenosine monophosphate. Since this single subunit binds to the other subunits, we hypothesise that the sites where this binding occurs may look like a ligand-binding pocket. We therefore believe the poor performance may be due to EquiBind mistaking these protein-protein binding sites for a protein-ligand binding pocket, so it therefore erratically places the ligands in a number of different pockets.

Similarly, *Tryptophan synthase alpha chain* is also a single subunit of a larger complex. Its performance is not as good as AutoDock Vina, however, not as bad as the previous targets - it has systematic shift in the positive x-direction of around 4Å, which may be for similar reasons to *5'-AMP-activated protein kinase catalytic subunit A*.

tRNA (guanine-N1-)-methyltransferase clearly performs very poorly compared to AutoDock Vina. Unlike the targets described above, we see no obvious explanation for this.

From the test set, *GTPase KRas* performs poorly. Like *PA-I galactophilic lectin*, it is also a homodimer, so we expect poor performance, however, it is very poor in comparison to *PA-I galactophilic lectin*, which may be because it is in the test set.

Finally, we have *ABC transporter, periplasmic substrate-binding protein*, which performs better than the *GTPase KRas*, but still systematically shifts the ligands away from the ground truth, most notably by about 3Å in the y-direction. This is quite a large target which may make it more difficult. However, it is also not trained on, so this may go some way towards confirming our hypothesis.

Having talked about the mean displacements, we must also discuss the standard deviations of the displacements. We note that EquiBind consistently has higher standard deviations for all targets, except *Calmodulin-domain protein kinase 1*, which is comparable to AutoDock Vina. This implies that, other than on easy targets in its training set, EquiBind is very sensitive to the initial conformation. We discuss a possible solution to this issue in Chapter 5.

In conclusion, we are unable to confirm or deny our hypothesis based on this experiment. We believe that this experiment has highlighted the difficulties inherent to blind docking, and feel that three of the targets chosen for the training set would present challenges for any blind docking software, and so don't make a good comparison with AutoDock Vina, which performs site-specific docking. The only 'easy' target from the training set - *Calmodulin-domain protein kinase 1* - performed as well as AutoDock Vina, and considerably better than any targets in the test set. This indicates that our hypothesis that EquiBind performs much better on targets it has seen could still hold. However, a larger scale study with a more targets is needed to confirm or deny this.

4.2 Redocking Generated Conformers

In this experiment we extend the work done in Section 4.1. In that experiment we presented the crystal poses as input for AutoDock Vina. This makes the task easier for AutoDock Vina: even though it performs flexible docking, its initial search space is nearer to the correct conformation.

In contrast, we now generate random initial conformers which are docked with AutoDock Vina. We then take the output poses and redock them with EquiBind again. This time, we expect AutoDock Vina to have a larger standard deviation and bigger displacements than before. However, once again, we expect that it will have no systematic shifts in any one direction. Like before, we hypothesise a difference in EquiBind's performance on its test and training set, in that the test set may have systematic shifts while the training set should not (although we also expect these shifts for the three difficult targets described above).

We add an additional hypothesis in this experiment: *EquiBind will produce poses with a higher standard deviation across all targets when compared to the results in Section 4.1.2, as it is being given a higher variability in input conformations.* The results of this experiment will confirm or deny our belief that EquiBind is overly sensitive to the input conformation.

4.2.1 Experiment design

Our experimental setup is identical in most respects to the design described in 4.1.1. We use the same targets and ligands. The only difference is that instead of taking mol2 structures directly from the PDBbind database, we use the SMILES strings for the exact same molecules provided by PDBbind. We then converted these SMILES strings into 3D conformers using the following Open Babel command:

```
obabel ligand.smi -O ligand.mol2 --gen3d --best --canonical --conformers
--weighted --nconf 50 --ff GAFF
```

The options here specify that it uses the Generalized Amber Force Field [56] (GAFF) to generate conformers. It uses a weighted rotor search algorithm to find the lowest energy conformers. It uses the slowest search speed to create high quality conformers. Additionally, it creates 50 low energy conformers, of which we use the best in our experiment.

These MOL2 files are then put through exactly the same pipeline as before: we generate PDBQT files for the ligand and target, dock them with AutoDock Vina, and then redock them with EquiBind. We present exactly the same metrics as before in our results section.

Additionally, for each molecule docked with AutoDock Vina, we calculate the Kabsch RMSD relative to the ground pose. The Kabsch RMSD calculates the minimum possible RMSD between the docked pose and the ground truth pose after a rotation and translation. It does this by first applying the Kabsch [25] algorithm, then calculates the RMSD which is the root of the mean of the squared displacements between the positions of all atoms in the ground truth pose with their corresponding position in the docked ligand. This provides a metric describing the difference between two different conformations of the same molecule.

We calculate the mean Kabsch RMSD for all poses for all ligands per target, which gives a single number measuring the variability of conformations output by AutoDock Vina. We do this for both our results here, and for the results of docking the crystal poses in the previous experiment. This informs our discussion of EquiBind's performance, as we use this conformation variability measure to assess EquiBind's sensitivity to initial conformations.

4.2.2 Results

Again both *3-phosphokimate 1-carboxyvinyltransferase* and *Arginase-1* had considerable processing issues, presenting broken molecules and fewer than nine poses, and so we discard these two targets.

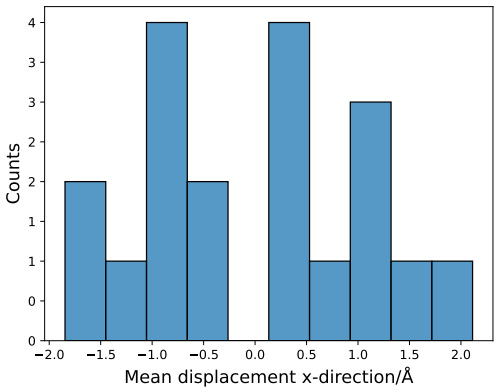
In Table 4.6 we show the mean Kabsch RMSD per target for the outputs of AutoDock Vina, run on both crystal poses and generated conformers.

Target	Generated	Crystal
Calmodulin-domain protein kinase 1	1.411	1.354
PA-I galactophilic lectin	1.925	1.633
5'-AMP-activated protein kinase catalytic subunit A	1.766	1.424
Tryptophan synthase alpha chain	1.621	1.584
tRNA (guanine-N1-)-methyltransferase	1.547	1.634
GTPase KRas	1.613	1.646
ABC transporter, periplasmic substrate-binding protein	1.625	1.597

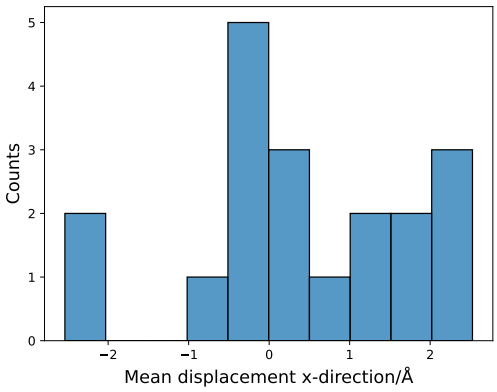
Table 4.6: Mean Kabsch RMSD per target/Å

We present here the same two cherry-picked targets as before. For the rest of our results, see Appendix C.

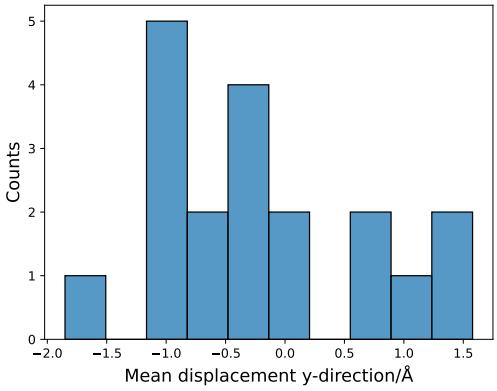
Calmodulin-domain protein kinase 1



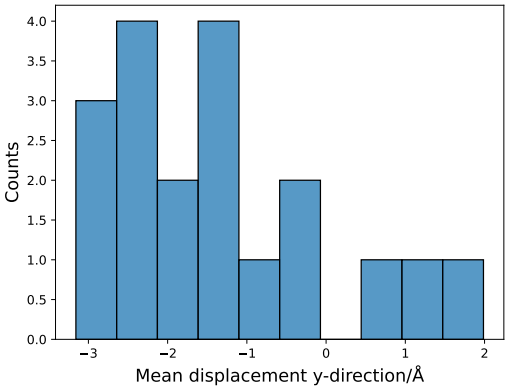
(a) Docked



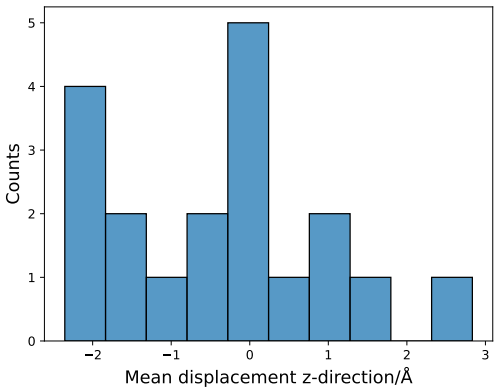
(b) Redocked



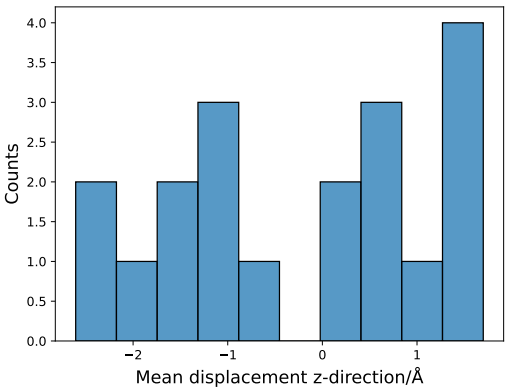
(c) Docked



(d) Redocked

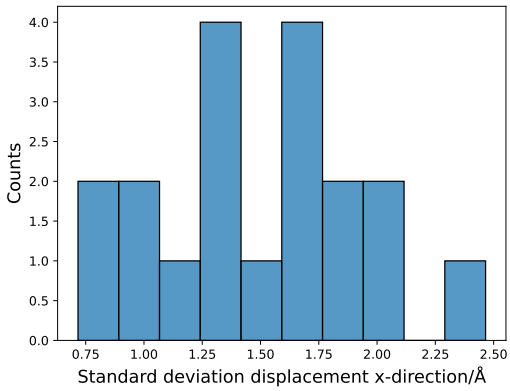


(e) Docked

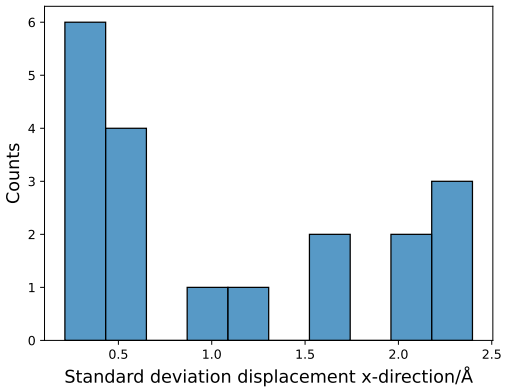


(f) Redocked

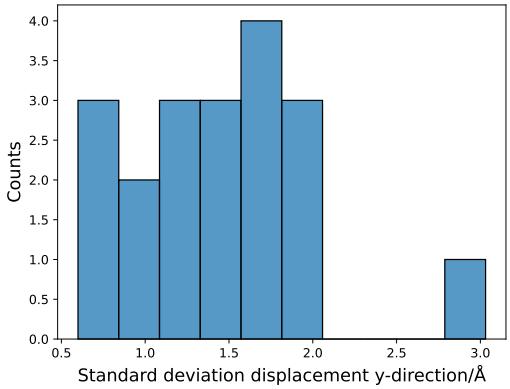
Figure 4.5: Calmodulin-domain protein kinase 1 mean displacements



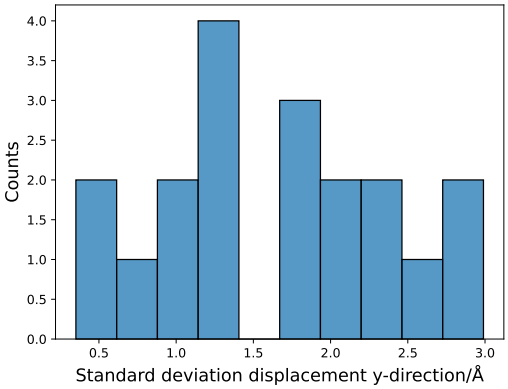
(a) Docked



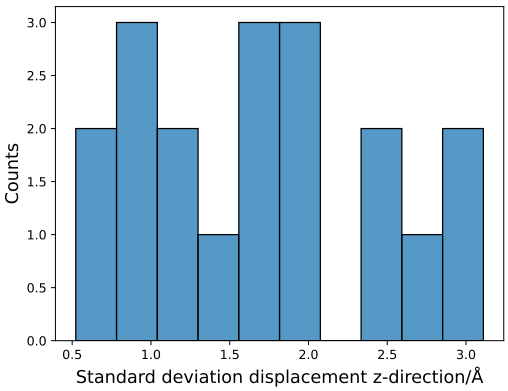
(b) Redocked



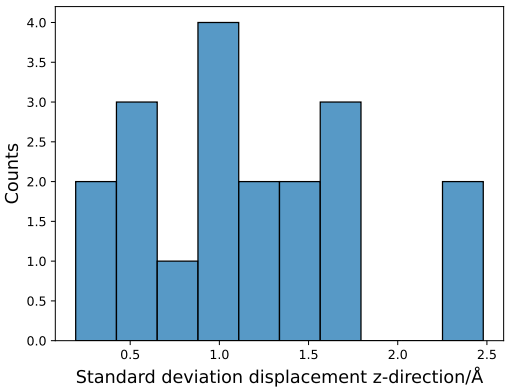
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure 4.6: Calmodulin-domain protein kinase 1 standard deviation displacements

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-0.051	-0.186	-0.345	1.486	1.447	1.716
std	1.142	0.941	1.411	0.457	0.561	0.754
min	-1.847	-1.851	-2.354	0.717	0.599	0.523
25%	-0.994	-0.867	-1.624	1.179	1.062	1.116
50%	0.162	-0.271	-0.201	1.464	1.435	1.746
75%	0.924	0.333	0.437	1.764	1.779	2.217
max	2.113	1.578	2.834	2.466	3.031	3.110

Table 4.7: Calmodulin-domain protein kinase 1 docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.344	-1.309	-0.240	1.082	1.651	1.166
std	1.407	1.435	1.394	0.818	0.784	0.627
min	-2.537	-3.158	-2.607	0.213	0.351	0.194
25%	-0.432	-2.322	-1.174	0.420	1.116	0.711
50%	0.254	-1.475	0.158	0.636	1.721	1.026
75%	1.413	-0.677	0.829	1.800	2.202	1.513
max	2.525	1.986	1.700	2.397	2.990	2.480

Table 4.8: Calmodulin-domain protein kinase 1 redocked metrics/Å

Test set

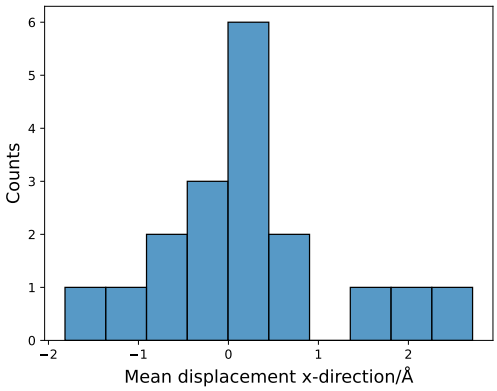
GTPase KRas

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.174	0.145	0.225	1.283	1.526	2.157
std	1.080	1.244	1.908	0.900	0.769	1.399
min	-1.816	-2.899	-3.991	0.399	0.595	0.483
25%	-0.311	-0.259	-0.636	0.633	1.043	1.076
50%	0.036	0.381	0.028	0.856	1.307	1.553
75%	0.566	0.914	1.262	2.016	1.609	3.513
max	2.713	2.020	2.859	3.061	3.221	4.463

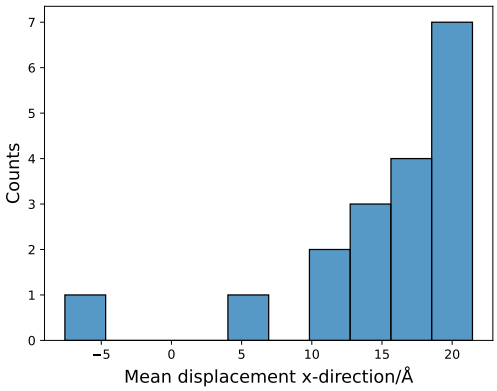
Table 4.9: GTPase KRas docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	15.259	-7.112	-12.353	15.203	6.960	11.666
std	6.986	21.972	12.242	4.280	20.632	11.536
min	-7.591	-94.889	-59.505	6.379	0.278	0.843
25%	14.337	-3.132	-11.577	13.532	0.873	8.358
50%	17.262	-2.686	-10.214	16.283	2.580	9.645
75%	19.383	-0.655	-8.848	18.278	3.012	10.930
max	21.446	1.504	-0.820	20.223	89.463	56.111

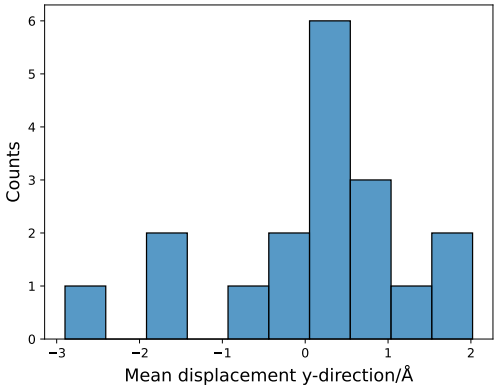
Table 4.10: GTPase KRas redocked metrics/Å



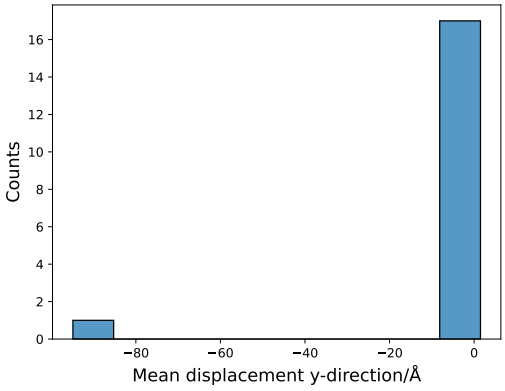
(a) Docked



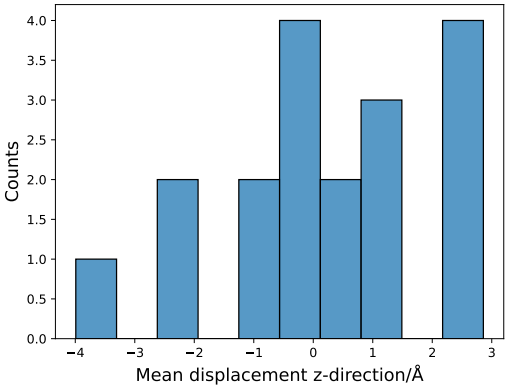
(b) Redocked



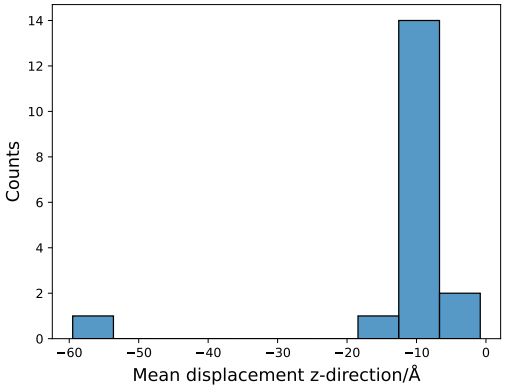
(c) Docked



(d) Redocked

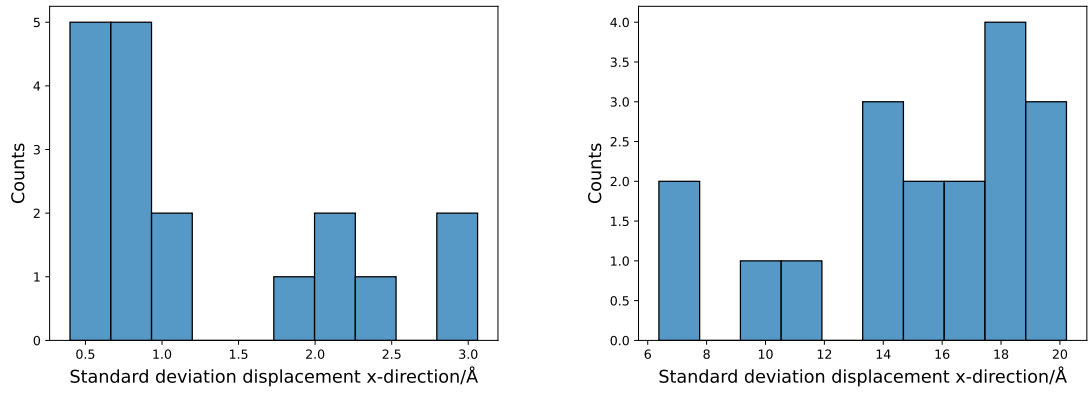


(e) Docked



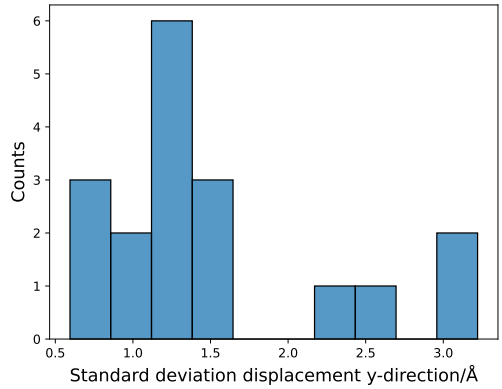
(f) Redocked

Figure 4.7: GTPase KRas mean displacements

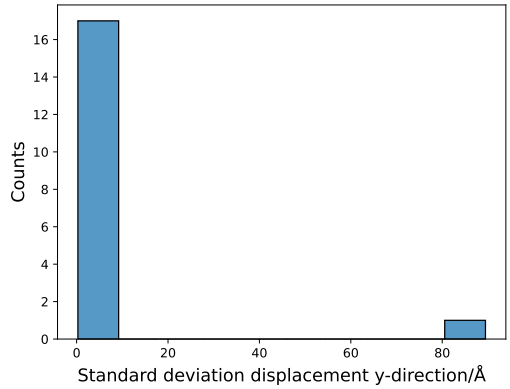


(a) Docked

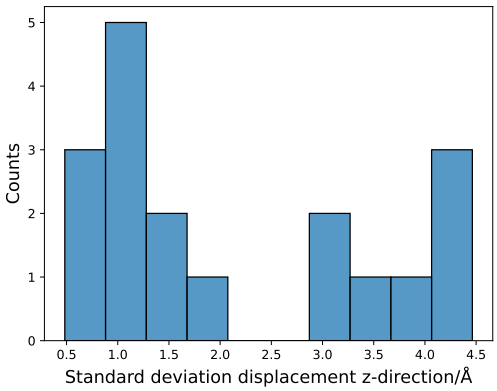
(b) Redocked



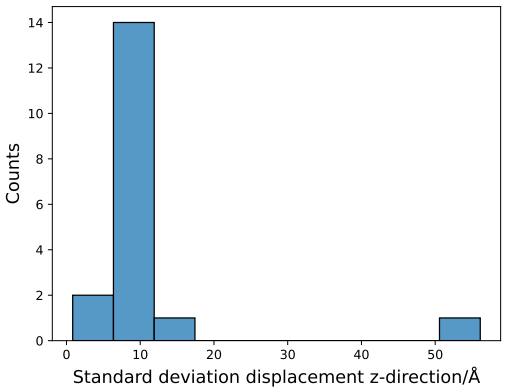
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure 4.8: GTPase KRas standard deviation displacements

4.2.3 Discussion

The results in Table 4.6 show a smaller difference in Kabsch RMSD between the docked crystal poses and the docked generated poses than we expected. Indeed, only *PA-I galactophilic lectin* and *5'-AMP-activated protein kinase catalytic subunit A* show a significant enough difference for us to deem the redocked generated poses worthy of detailed discussion.

We note that the redocked generated poses for *PA-I galactophilic lectin* are considerably more off-centre and have considerably higher standard deviations compared to their redocked crystal poses. However, this result is not backed up by *5'-AMP-activated protein kinase catalytic*

subunit A, which has broadly similar performance for both the redocked crystal poses and redocked generated poses.

The rest of the targets all perform fairly similarly, in both experiments, or only slightly worse in the generated conformers experiment, except for *ABC transporter, periplasmic substrate-binding protein*, which performs considerably worse in the generated conformers experiment. This is a very interesting result, because this target performed fairly well in the redocking crystal poses experiment. Also, AutoDock Vina performs very well on the generated poses - it has low standard deviations and almost zero offset. However, it then performs terribly when these poses are redocked with EquiBind. It seems to sometimes place ligands in the correct pocket, and then sometimes shifts the ligands around 100Å in the negative y- and z-direction. This means it is probably moving the ligand across the entire protein to a different pocket. However, it may be doing something else entirely as this behaviour is extreme, so we are tempted to call this result an experimental outlier.

In conclusion, we have failed to demonstrate that our hypothesis holds, because the results of docking generated conformers with AutoDock Vina didn't give the larger variation in conformations we were hoping for. However, we have shown that EquiBind does sometimes have quite unusual behaviour, and that this varies greatly by target. We therefore argue that the usefulness of EquiBind is very target-dependent. In a commercial setting, it may take some considerable expertise to determine when it is an appropriate docking tool. It seems to perform very well on simple targets with only one pocket, but perhaps less well on trickier targets.

4.3 Antibiotics

Following on from our previous conclusion, in this experiment we aim to demonstrate the benefits and pitfalls of using EquiBind in a realistic drug discovery pipeline.

Our target in this experiment is the MurD ligase, from the open source antibiotics challenge. The ligands are the 75 hits, described in Section 2.3, and listed explicitly in Appendix D.

We once again generate initial conformers for the 75 ligands, dock them to the allosteric pocket defined in the challenge using AutoDock Vina, then redock them using EquiBind. As discussed in Section 2.3, this target is an enzyme and has two pockets - the active site which performs the enzyme's catalytic function, and a hypothesised allosteric site which can regulate its activity. Based on our previous experiment, it seems that EquiBind has a tendency to shift ligands to the most explicit pocket. We therefore hypothesise that: *EquiBind will shift all docked ligands from the allosteric site to the active site*. In particular, this would make it unsuitable for use in this drug discovery challenge.

4.3.1 Experiment design

Our procedure is identical to that undertaken in Section 4.2.1. The only difference is that we test two different methods for generating the initial conformers from the SMILES strings. In addition to the method used before, which we refer to as GAFF and restate now, we additionally test another method which we refer to as PH74.

GAFF:

```
obabel ligand.smi -O ligand.mol2 --gen3d --best --canonical --conformers
--weighted --nconf 50 --ff GAFF
```

PH74:

```
obabel ligand.smi -O ligand.mol2 --gen3d best -p 7.4 --canonical
```

The options here specify that it uses the default forcefield Merck Molecular Force Field [19] (MMFF94). Additionally, it specifies that the molecules are protonated such that the pH equals 7.4.

We make one further modification to our experimental approach: due to our strong suspicion that EquiBind is shifting all ligands to the active site, it would be inappropriate to measure displacements relative to the ground truth in the allosteric pocket. We therefore redefine the ground truth to be the lowest energy pose predicted by AutoDock Vina, redocked with EquiBind. We then measure the mean and standard deviation displacements of the remaining poses relative to this ground truth.

4.3.2 Results

First of all as a visualisation of this process we show in Figure 4.9 the lowest energy pose for the first ligand before and after redocking. We see for this particular ligand EquiBind is shifting its position quite considerably.

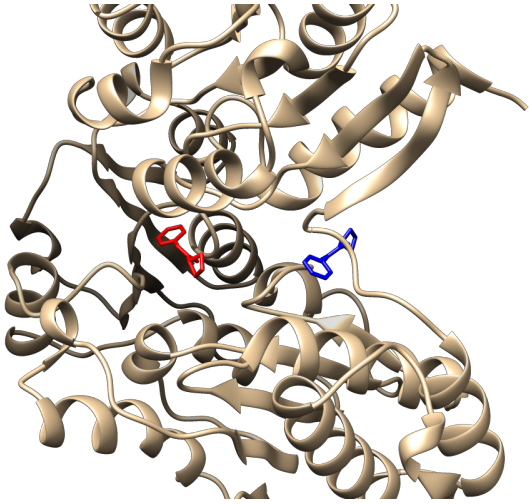
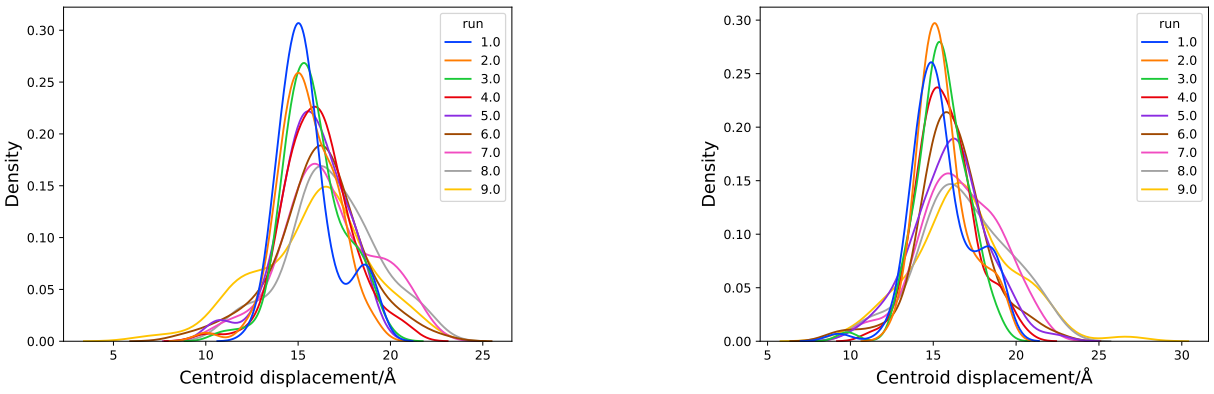


Figure 4.9: MurD ligase with docked (blue) and redocked (red) ligand



(a) GAFF

(b) PH74

Figure 4.10: KDE plots showing the distribution of displacements of the ligand centroids

In Figure 4.10 we show kernel density estimator (KDE) plots representing the distribution of the displacements of the centroids before and after redocking, for both datasets. Separate distributions are calculated for the top scoring original pose through to the ninth highest scoring original pose. Note a KDE plot is very similar to a histogram - while a histogram measures the number of data points in each bin, a KDE plot simply smooths the histogram out using a kernel to give an estimate of what the underlying distribution may be. Note the frequency is over all 75 ligands.

In Table 4.11 we also show a number of metrics calculated from this plot. For simplicity we combine all nine runs for these metrics.

	GAFF	PH74
mean	15.988	16.010
std	2.152	2.197
min	7.116	9.113
25%	14.752	14.735
50%	15.867	15.816
75%	17.165	17.336
max	22.447	26.619

Table 4.11: Centroid displacement/Å

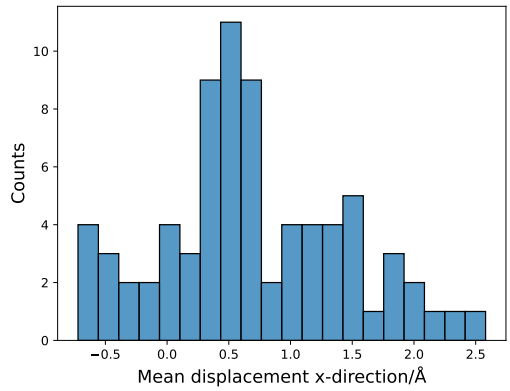
In Figure C.10 and 4.12 we show the x, y and z displacement histograms of the eight ligand poses relative to the ground truth pose, for both the GAFF and PH74 generated conformers. Note the count is taken over all 75 ligands. In Table 4.12 and 4.13 we show a number of metrics calculated from these plots.

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.698	2.617	0.185	1.619	4.326	2.156
std	0.755	2.386	1.527	0.764	1.833	0.972
min	-0.722	-1.057	-5.476	0.252	0.524	0.570
25%	0.291	0.909	-0.741	1.062	3.870	1.265
50%	0.559	2.248	0.498	1.754	4.729	2.327
75%	1.217	4.193	1.300	2.202	5.690	2.982
max	2.580	10.778	3.098	3.040	7.422	4.048

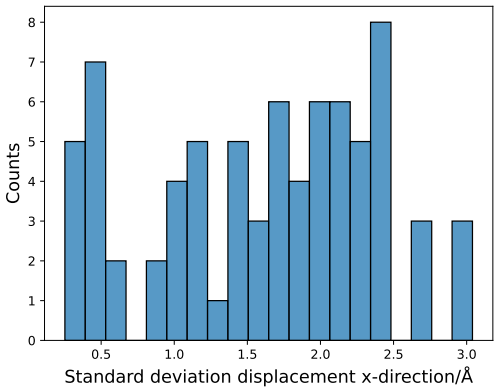
Table 4.12: GAFF displacements relative to ground truth/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.596	1.919	0.328	1.620	4.023	2.100
std	1.083	2.925	1.599	0.759	1.954	0.902
min	-4.477	-10.317	-4.217	0.086	0.289	0.524
25%	0.017	0.536	-0.292	0.988	3.045	1.414
50%	0.564	2.020	0.778	1.772	4.537	2.058
75%	1.278	3.431	1.173	2.244	5.489	2.706
max	2.985	8.318	5.583	3.155	7.626	4.256

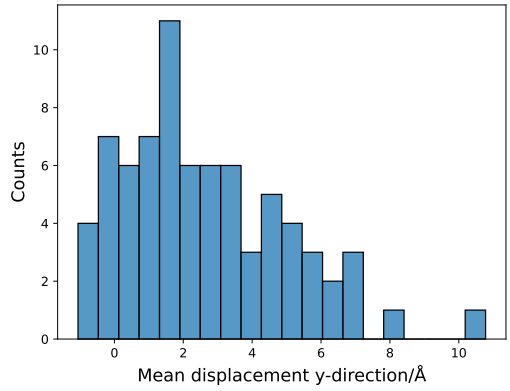
Table 4.13: PH74 displacements relative to ground truth/Å



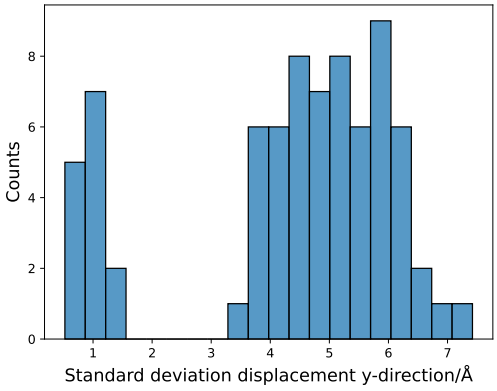
(a)



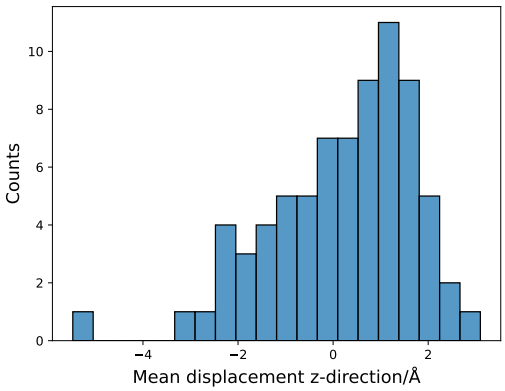
(b)



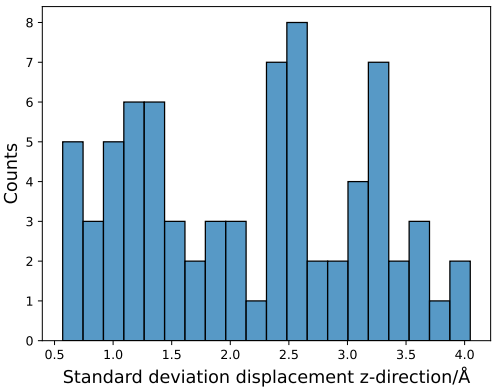
(c)



(d)



(e)



(f)

Figure 4.11: GAFF displacements relative to ground truth

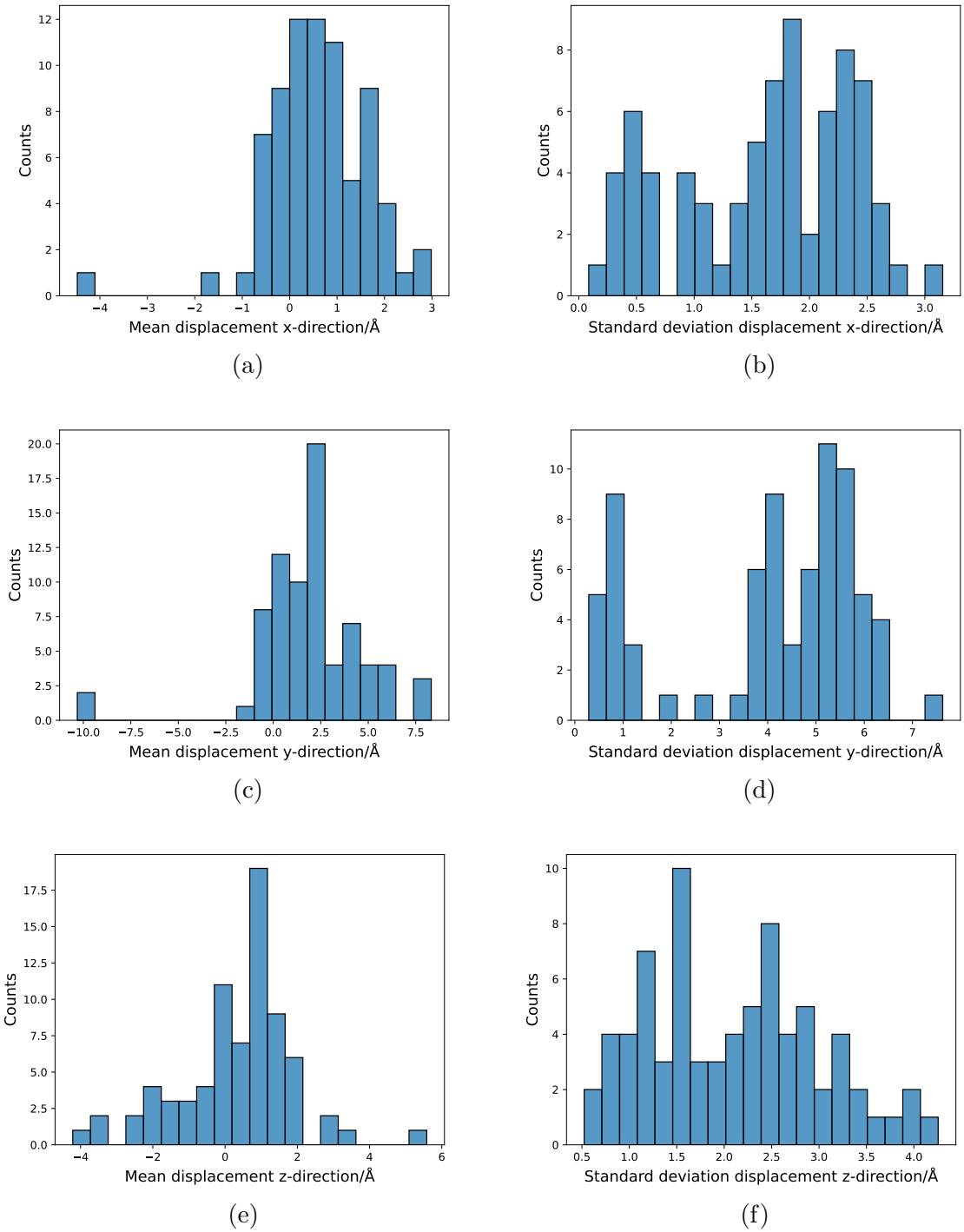


Figure 4.12: PH74 displacements relative to ground truth

4.3.3 Discussion

From these results it is clear that EquiBind is shifting all the ligands from the allosteric site to the active site. The means of the displacements of the centroids before and after docking is around 16Å,, implying it is moving the ligands to a different pocket. Additionally, we note that the standard deviations of the displacements relative to the ground truth are fairly small, implying that EquiBind is finding a single pocket (we presume the active site), to which it is shifting all the ligands.

We also note that the mean displacements are again off-centre. We do not make any con-

clusions from this result, however. This is because the lowest energy conformation output by AutoDock Vina is the lowest energy for the allosteric site, not the active site, so we cannot conclude this same conformation would give the lowest energy pose when redocked with EquiBind into the active site.

Finally, we note that there is very little difference in performance between the GAFF and PH74 experiments, meaning the method by which we generate initial conformers doesn't seem to affect the output of redocking very much.

In conclusion, while EquiBind can very useful in a blind docking scenario, or when the receptor has only one pocket, our results unfortunately mean that EquiBind's usefulness is limited in the case where a target has more than one known pocket. In this case, we conclude that classical docking software which can perform site-specific docking is a more appropriate tool.

4.4 EquiBind-score

In this experiment we test our second hypothesis: *EquiBind learns implicit 3D fingerprints in its layers that can be generalised to other prediction tasks, including binding affinity prediction.*

We present three related models that modify the original EquiBind architecture so that it predicts the binding affinity instead of the docked pose. We call this new model EquBind-score. We train and test this model on the PDBbind dataset. The results of this experiment indicate what EquiBind is learning in its different layers.

The code for this experiment can be found at:

https://github.com/TBFS6/equibind-score

4.4.1 Experiment design

Architecture

Recall the original EquiBind architecture depicted in Figure 3.4. For EquiBind-score, we extract the final ligand and receptor graphs after the IEGMN layers, before they get fed to the multi-head attention mechanism. We discard the transformed coordinates Z and Z', so we make use only of the H and H' layers. Note that we keep the original EquiBind weights frozen before this extraction - they do not get retrained.

We then treat these hidden feature graphs as 3D molecular fingerprints which we use as input into our EquiBind-score model. Recall that the IEGMN layers in EquiBind are message-passing layers, so information is passed between the ligand and receptor graphs. Our hope is that enough information is passed between these two graphs that these fingerprints contain descriptions of the protein-ligand interactions which can be used to effectively predict the binding affinity.

We then apply a graph attention network [54] (GAT) to these fingerprint graphs. This is inspired by the results of Moesser et al. [34] where a GAT network performed best at predicting binding affinity on their protein-ligand interaction graph, as described in Section 3.5. Recall from Figure 3.2 and Equation 3.4 the construction of a single GAT layer.

We present three closely related models: one that makes use of just the ligand graph, one that uses just the receptor graph, and one that uses both graphs.

Our network is implemented using PyTorch [40] and DGL [57]. For all models we apply two GAT layers to the graphs: the first layer goes from 64 features per node to 64 features per node, with 10 multi-attention heads, then a ReLU activation function. The 10 attention heads

are summed, then the second layer goes from 64 hidden features to 128 features, before another ReLU activation is applied, and a maxpool function is applied across the graphs to return a linear vector with 128 features.

For the separate receptor and ligand models this vector is then passed through a linear layer to a vector with 128 hidden features, a ReLU activation, then a final linear layer to a single scalar output representing the binding affinity. For the model using both graphs, the two outputs of the individual GAT layers are appended to give a vector with 256 features, which is then passed through a linear layer to a vector with 128 features, a ReLU, and another linear layer to give a scalar output.

Dataset

We use PDBbind to train and test our model. In particular, we use the exact same train/validation/test split used in the original EquiBind. This is because the EquiBind layers used to exact the fingerprints are trained on this split, so use of a different split would contaminate the test set, as parts of the full model could have already been trained on complexes in the final test split. There are 16,379 complexes in the training set, 968 in the validation set, and 363 in the test set.

Hyperparameters and training regime

All of our hyperparameters are tuned by hand, by evaluating performance on the validation set. We use the same hyperparameters for all three models.

We train the model using ADAM [26] as an optimizer, with a learning rate of 0.0005. We perform the optimization on batches of size 100, and iterate through all training batches 10 times. We use a mean square error loss, with the pK values provided by PDBbind as a target. We use a feature dropout of 0.05 on the GAT layers, and a dropout of 0.2 on the linear layers.

4.4.2 Results

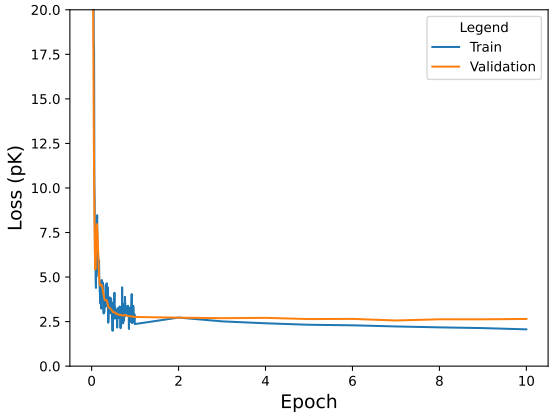
In Table 4.14 we show both the root mean square error (RMSE) and the Pearson correlation coefficient, on the test set. The Pearson correlation coefficient is a measure of linear correlation ranging from -1, indicating a perfect negative correlation, to +1, indicating a perfect positive correlation, with 0 indicating no correlation. Note that a Pearson correlation coefficient of around 0.6 indicates a moderate positive correlation.

Model	RMSE (pK)*	Pearson
Ligand only	1.468	0.609
Receptor only	1.647	0.577
Ligand and receptor	1.609	0.618

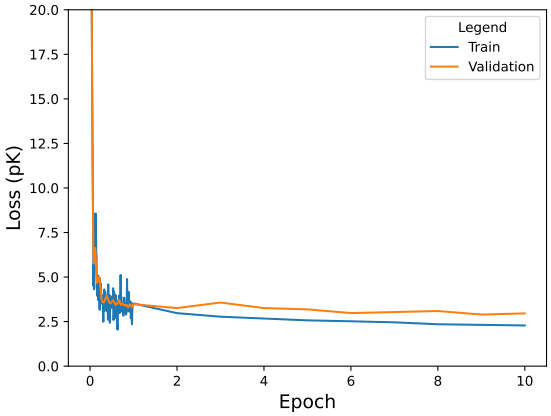
Table 4.14: RMSE and Pearson correlation coefficient for the three models

In Figure 4.13 we show the losses as a function of training iteration number. Note that we explicitly plot the losses for each batch through the first epoch, but since this is computationally demanding we only computed the validation loss at the end of each full iteration thereafter, and hence also only plotted this for the training set.

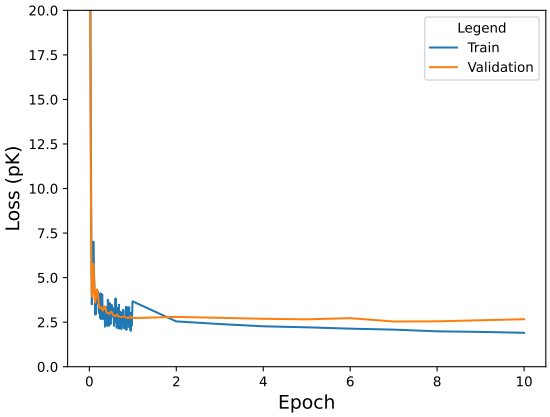
*We put pK in brackets to show that this is a pK value. pK is not a unit, since a pK affinity is the logarithm of a value with unit M



(a) Ligand only



(b) Receptor only



(c) Receptor and ligand

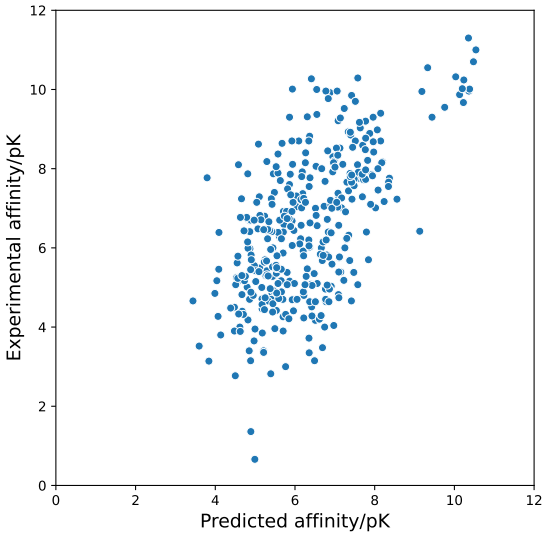
Figure 4.13: Training and validation losses

To visualise these results, we additionally show scatter plots of the test set experimental values vs predictions in Figure 4.14.

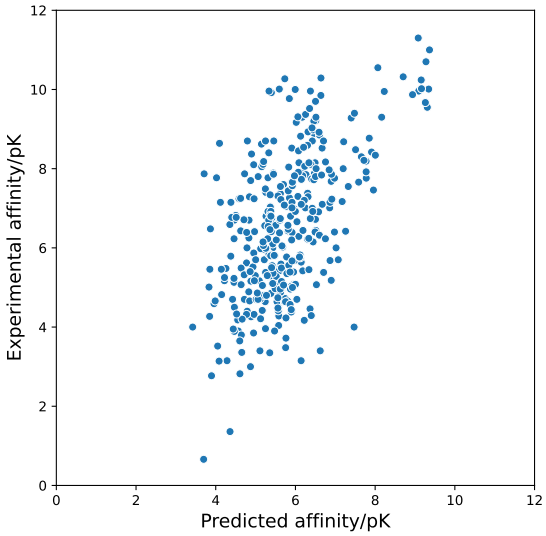
4.4.3 Discussion

The results for all three models are moderately good. Although a direct comparison with Moesser et al. [34] is not possible due to the use of a different test set, we note that these three models perform almost as well in RMSE as the various graph neural networks which they applied to their protein-interaction ligand graphs. This result is impressive because the protein-ligand interaction graphs require an already docked pose, whereas our model only requires the undocked protein and ligand structures. However, they perform slightly more poorly when measured by the Pearson correlation coefficient.

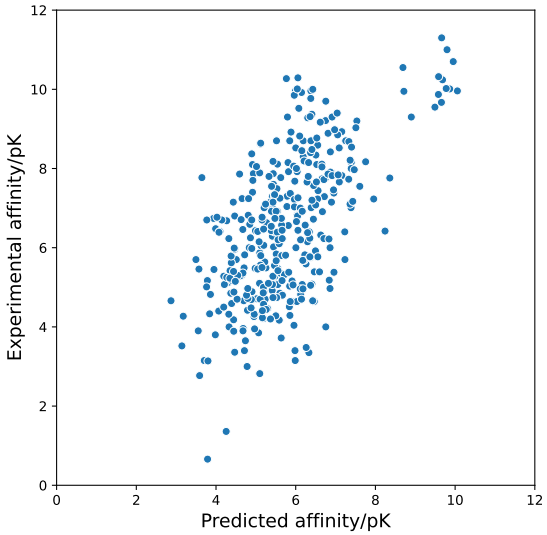
Additionally, we argue that Moesser et al. [34] tested their models on an easier test set - the CASF dataset (the PDBbind core set) - which was not used by Stärk et al. [50] because its high quality also means it typically provides easier complexes to predict than would be encountered with the EquiBind timesplit that we used.



(a) Ligand only



(b) Receptor only



(c) Receptor and ligand

Figure 4.14: Experimental affinities vs predicted affinities (pK)

We believe the difference in performance between the three models is almost entirely due to noise. From Figure 4.13, we see a moderate amount of noise in the loss, so with slightly different hyperparameters or a different number of training epochs the order of performance of the models could change.

Concluding that the three models have very equivalent performance is a very interesting fact. It shows that EquiBind is passing a considerable amount of information between the receptor and ligand graphs in the IEGMN layers, such that after the IEGMN layers both graphs contain a similar amount of information about the final docked pose. This goes some way towards opening up the 'black box' of the large neural network that makes up EquiBind.

There is a complication when comparing our method with PLIGs. Since PLIGs require a docked pose, it is not necessary for them to learn to predict complexes that simply do not bind

to each other at all. In our case, however, we can feed in such complexes to the model, so it is important that it can correctly predict when a complex will not bind.

4.5 Decoys

In this experiment we test EquiBind-score on an alternate test set composed of both decoys and actives. The test set is the diverse set provided by DUD-E, described in Section 2.4. The architecture of EquiBind-score is designed to always predict an affinity - it cannot output 'does not bind'. However, it is trained on a large variety of complexes, some weakly binding, and some strongly binding. We therefore hope that its training set contains enough information for it to predict a very low binding affinity for decoys, implying that they may not bind at all. We hypothesise that: *EquiBind-score predicts lower binding affinities for decoys than it does for actives.*

4.5.1 Experiment design

For each of the eight targets in the DUD-E diverse set, we randomly sample 10 actives. For each active, we then randomly sample 10 decoys. We then run all the actives and decoys through EquiBind-score with their associated targets. We use the EquiBind-score mode that uses both the ligand and receptor graph.

For each active, we calculate the mean and standard deviation of the 10 decoy pK values, which we then compare with the predicted active pK.

4.5.2 Results

We encountered a number of processing issues running the decoys through EquiBind-score. This is not unexpected as decoys are generated molecules, and while DUD-E provides some filtering to ensure they are valid, it is still expected that they will encounter more issues when being run through any docking software. In the event of an issue docking a decoy, we discarded the active and all other decoys associated with that molecule.

For four of the targets every active had at least one decoy that encountered processing issues. These targets were: CP3A4, CXCR4, HIVRT and KIF11.

For the remaining targets we now show the active predicted affinity, along with the mean and standard deviation of the decoy affinities.

Ligand	Active pK	Mean decoy pK	Std decoy pK
1	7.517	5.958	1.152
2	7.191	6.193	0.426
3	7.750	5.505	0.683
4	7.294	6.646	1.045
5	8.668	6.223	0.601
6	7.518	6.005	0.319
7	7.230	5.347	0.721
8	5.689	5.960	0.387
9	7.527	6.328	0.672
10	6.411	6.335	0.695

Table 4.15: AKT1 affinity predictions

Ligand	Active pK	Mean decoy pK	Std decoy pK
1	4.841	4.676	0.681
2	3.622	3.600	1.020
3	4.925	4.146	1.267
4	4.436	4.939	0.418
5	4.461	4.069	1.614

Table 4.16: AMPC affinity predictions

Ligand	Active pK	Mean decoy pK	Std decoy pK
0	7.969	6.434	0.746
1	7.186	8.125	0.918
2	7.804	7.914	0.426
3	8.539	7.019	0.809
4	6.601	7.670	0.945
5	7.926	6.800	1.656

Table 4.17: GCR affinity predictions

Ligand	Active pK	Mean decoy pK	Std decoy pK
0	6.540	6.530	0.893
1	6.275	6.629	0.538
2	7.305	6.624	0.691
3	7.219	6.525	0.702
4	8.221	6.824	0.662
5	6.926	6.708	0.687
6	6.980	6.453	0.754

Table 4.18: HIVPR affinity predictions

In Figure 4.15 we show KDE plots representing the distribution of active affinities minus decoy affinities, computed over all decoys.

4.5.3 Discussion

Our model performs very poorly on decoys. On average, it predicts that decoy affinities are about as high as the active affinities, or only very slightly lower.

This undermines the success of this model measured in the previous experiment. A model that predicts the same binding affinity for a decoy as it doss the associated active is clearly using global properties of the ligand to make predictions, as opposed the the 3D chemical information, which is more informative.

This implies that the receptor and ligand graphs extracted from EquiBind for EquiBind-score do not hold up as robust 3D molecular fingerprints.

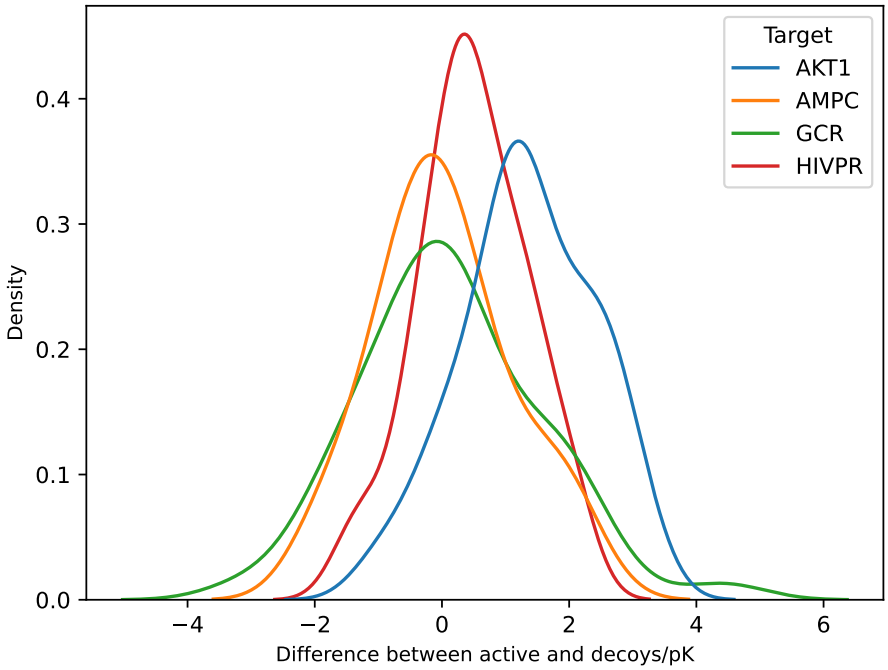


Figure 4.15: KDE plots of the difference between the active and decoy predictions

However, neither EquiBind nor EquiBind-score have been trained on decoys, so it is possible that this result is not that indicative of the architecture's ability to detect decoys if trained on them.

We believe this result implies that EquiBind itself cannot recognise decoys. EquiBind's architecture is designed so that it will always place the ligand somewhere - it has no option to output 'does no dock', and has not been trained on any decoys. However, prior to this experiment, we still believed some information in its layers could be encoding that a molecule doesn't dock. This has turned out to be false.

Chapter 5

Discussion and Future Work

5.1 Discussion

We now return to our two primary hypotheses, and evaluate what implications our experiments have for them.

The first three experiments explored our first hypothesis that *EquiBind has comparable performance to classical docking methods on targets in its training set, but is not a useful docking method on unseen targets.* Our redocking experiments neither confirm nor deny this hypothesis, due to the tricky nature of the targets we chose from the training set.

Our work, however, does indicate many limitations of EquiBind and blind docking more generally. In particular, we show a large target-dependence of the performance of EquiBind. For some targets, it appears that EquiBind has a strong preference for a single pocket, and will place all ligands there regardless of whether they may bind better in a different location. For example, we showed this in the antibiotics experiment where it was clearly unable to recognise the existence of an allosteric site. For other targets, however, it seems to behave more sporadically, placing different input conformers of the same ligand in completely different locations.

Both of these behaviours are undesirable - a good docking algorithm should have little dependency on the input conformer, as it should have the ability to explore a number of different conformations and pockets to find what fits best. Additionally, the pocket location it finds should be dependent on the ligand, since in nature different ligands have preferences for different pockets.

EquiBind does perform well on its simple test set from PDBbind, and especially well when using the crystal ligands as input. However, as our tests get closer to how EquiBind would be used in a realistic virtual screening scenario, its performance decreases.

These behaviours could be implemented with a more diverse training set, and in particular a more diverse set of targets. Since some targets really do have only one pocket, and some have more than one, it is important for any ML model to be able to recognise this on an unseen target, as this greatly affects its performance. Additionally, we propose a modification that could reduce the effect of dependency on the input conformer in Section 5.2.

Additionally, we tested our hypothesis that *EquiBind learns implicit 3D fingerprints in its layers that can be generalised to other prediction tasks, including binding affinity prediction.* In our EquiBind-score experiment, we showed that EquiBind learns enough information in its receptor and ligand graphs to predict binding affinities with moderately good accuracy, and

that the final receptor and ligand graphs contain very similar information. However, we also showed that this model performs very poorly on decoys, implying that EquiBind itself is not good at recognising decoys, and perhaps relies too much on global ligand properties to make predictions and not enough on structural information.

A number of the drawbacks of using EquiBind that we have demonstrated are, in fact, drawbacks of using blind docking more generally. We therefore conclude on a positive note about EquiBind. We restate that EquiBind's immense computational speedup when compared to classical docking methods, while achieving a similar performance, is an incredible advancement in the field. We believe that models like EquiBind, and particularly those that are based on geometric deep learning architectures, will become increasingly prominent. With any advancement, the first big leap often has many wrinkles that need ironing out, and it is these issues that we have attempted to uncover, so that future models can build on the success of EquiBind even further.

5.2 Future Work

Since we were unable to confirm or deny our first hypothesis based on our data selection, our starting point for future work would be to perform our two redocking experiments on a broader range of targets. In particular, given enough computational resources, we would like to perform the redocking experiment on the entirety of PDBbind. We could then aggregate the results across all targets included in the training or validation set, and the targets that are not in the training or validation set, to rigorously assess the difference in performance.

Additionally, it would be of interest to retrain the EquiBind-score model on a large set of decoys from DUD-E, along with some actives. We would need to unfreeze the original EquiBind layers in addition to the GAT layers. We would make a small modification to the architecture so that it first predicts a binary value, representing whether the complex binds or doesn't bind, and then if it predicts that it binds it can additionally predict an affinity. This experiment would test whether the problem in the decoys experiment is simply with the EquiBind training data, since PDBbind has no examples of complexes that do not bind, or whether it is an issue with the EquiBind architecture.

Ultimately, we believe a long-term goal of the field is to create a GDL-based model that can perform site-specific docking, that can output an accurate binding affinity and a docked pose, and has the option to output 'does not bind'. Additionally, the model should be robust to input conformers, or instead, could take ligand SMILES strings as input. The creation of such a model, if it has good performance and requires a similar compute time to EquiBind for inference, would have immense implications for virtual screening. Recent advancements such as Uni-Mol [59] show that GDL architectures are a promising area for further exploration in this field.

There are two additional experiments that we would like to explore, but did not have time to do so during this project. The first explores a hypothesis of ours about how to increase the robustness of EquiBind to the input ligand conformer. We would make a simple adaptation to EquiBind-score so that it additionally runs the original EquiBind model, and therefore outputs both a docked pose and a binding affinity. Our suggestion is then to generate a number of diverse input conformers with RDKit, say 100, and input them all into EquiBind for inference, then select the output with the lowest binding affinity. We could then perform a number of docking experiments, varying the number of input conformers, and measuring the displacements of the ligands, as a way of exploring further the sensitivity to the input conformer and assessing whether EquiBind-score can be used to reduce this sensitivity.

The second experiment involves applying meta-learning to EquiBind. In particular, we can view the prediction of the ligand docked pose as a different task for each target - i.e., we could train a different model for each target as is often the case for binding affinity prediction models. The goal of a meta-learning approach to docking would be to determine model weights that can be used to regularise training the model for different tasks. For example, the approach we would explore is to learn a robust set of model weights that work fairly well across a large number of targets (basically how EquiBind is currently trained). We would then explore the effect of training on a new target with some experimental data, using these weights as a starting point with very quick early stopping. We hypothesise that this approach could greatly improve EquiBind's performance on a target with limited experimental data, with some brief inexpensive retraining.

Bibliography

[1] URL: http://www.pdbbind.org.cn/download/pdbbind_2020_intro.pdf.

[2] URL: https://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html.

[3] URL: https://autodock.scripps.edu/wp-content/uploads/sites/56/2021/10/AutoDock4.2.6_UserGuide.pdf.

[4] Seth D. Axen et al. "A Simple Representation of Three-Dimensional Molecular Structure". In: *J. Med. Chem.* 60.17 (Sept. 2017), pp. 7393–7409. ISSN: 1520-4804. DOI: 10.1021/acs.jmedchem.7b00696. eprint: 28731335.

[5] Helen M. Berman et al. "The Protein Data Bank". In: *Nucleic Acids Res.* 28.1 (Jan. 2000), pp. 235–242. ISSN: 0305-1048. DOI: 10.1093/nar/28.1.235.

[6] G. Richard Bickerton et al. "Quantifying the chemical beauty of drugs". In: *Nat. Chem.* 4.2 (Jan. 2012), pp. 90–98. ISSN: 1755-4349. DOI: 10.1038/nchem.1243. eprint: 22270643.

[7] Regine S. Bohacek, Colin McMartin, and Wayne C. Guida. "The art and practice of structure-based drug design: A molecular modeling perspective". In: *Med. Res. Rev.* 16.1 (Jan. 1996), pp. 3–50. ISSN: 0198-6325. DOI: 10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6.

[8] Fergus Boyles, Charlotte M. Deane, and Garrett M. Morris. "Learning from the ligand: using ligand-based features to improve binding affinity prediction". In: *Bioinformatics* 36.3 (Feb. 2020), pp. 758–764. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz665.

[9] Michael M. Bronstein et al. "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges". In: *arXiv* (Apr. 2021). DOI: 10.48550/arXiv.2104.13478. eprint: 2104.13478.

[10] Beining Chen et al. "Evaluation of machine-learning methods for ligand-based virtual screening". In: *J. Comput.-Aided Mol. Des.* 21.1 (Jan. 2007), pp. 53–62. ISSN: 1573-4951. DOI: 10.1007/s10822-006-9096-5.

[11] Tiejun Cheng et al. "Comparative assessment of scoring functions on a diverse test set". In: *J. Chem. Inf. Model.* 49.4 (Apr. 2009), pp. 1079–1093. ISSN: 1549-9596. DOI: 10.1021/ci9000053. eprint: 19358517.

[12] Yung-Chi Cheng and William H. Prusoff. "Relationship between the inhibition constant (KI) and the concentration of inhibitor which causes 50 per cent inhibition (I50) of an enzymatic reaction". In: *Biochem. Pharmacol.* 22.23 (Dec. 1973), pp. 3099–3108. ISSN: 0006-2952. DOI: 10.1016/0006-2952(73)90196-2.

[13] Alberto Cuzzolin et al. "DockBench: An Integrated Informatic Platform Bridging the Gap between the Robust Validation of Docking Protocols and Virtual Screening Simulations". In: *Molecules* 20.6 (May 2015), pp. 9977–9993. ISSN: 1420-3049. DOI: 10.3390/molecules20069977.

[14] M. Dixon. "The determination of enzyme inhibitor constants". In: *Biochem. J.* 55.1 (Aug. 1953), pp. 170–171. ISSN: 0264-6021. DOI: 10.1042/bj0550170. eprint: 13093635.

[15] David S. Dummit and Richard M. Foote. *Abstract Algebra*. Chichester, England, UK: Wiley, July 2003, pp. 442, 446, 452–458. ISBN: 978-0-47143334-7.

[16] Jerome Eberhardt et al. “AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings”. In: *J. Chem. Inf. Model.* 61.8 (Aug. 2021), pp. 3891–3898. ISSN: 1549-960X. DOI: 10.1021/acs.jcim.1c00203. eprint: 34278794.

[17] Richard A. Friesner et al. “Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy”. In: *J. Med. Chem.* 47.7 (Apr. 2004), pp. 1739–49. ISSN: 0022-2623. DOI: 10.1021/jm0306430.

[18] Octavian-Eugen Ganea et al. “Independent SE(3)-Equivariant Models for End-to-End Rigid Protein Docking”. In: *arXiv* (Nov. 2021). DOI: 10.48550/arXiv.2111.07786. eprint: 2111.07786.

[19] Thomas A. Halgren. “Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94”. In: *J. Comput. Chem.* 17.5-6 (Apr. 1996), pp. 490–519. ISSN: 0192-8651. DOI: 10.1002/(SICI)1096-987X(199604)17:5/6<490::AID-JCC1>3.0.CO;2-P.

[20] Stephen Heller et al. “InChI - the worldwide chemical structure identifier standard”. In: *J. Cheminf.* 5 (2013), p. 7. DOI: 10.1186/1758-2946-5-7.

[21] Sheng-You Huang. “Comprehensive assessment of flexible-ligand docking algorithms: current effectiveness and challenges”. In: *Briefings Bioinf.* 19.5 (Sept. 2018), pp. 982–994. ISSN: 1477-4054. DOI: 10.1093/bib/bbx030.

[22] J. P. Hughes et al. “Principles of early drug discovery”. In: *Br. J. Pharmacol.* 162.6 (Mar. 2011), pp. 1239–1249. ISSN: 0007-1188. DOI: 10.1111/j.1476-5381.2010.01127.x.

[23] Gareth Jones et al. “Development and validation of a genetic algorithm for flexible docking11Edited by F. E. Cohen”. In: *J. Mol. Biol.* 267.3 (Apr. 1997), pp. 727–748. ISSN: 0022-2836. DOI: 10.1006/jmbi.1996.0897.

[24] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596 (Aug. 2021), pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2.

[25] W. Kabsch. “A solution for the best rotation to relate two sets of vectors”. In: *Acta Crystallogr. A* 32.5 (Sept. 1976), pp. 922–923. ISSN: 0567-7394. DOI: 10.1107/S0567739476001873.

[26] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *undefined* (2015). URL: https://www.semanticscholar.org/paper/Adam%3A-A-Method-for-Stochastic-Optimization-Kingma-Ba/a6cb366736791bcccc5c8639de5a8f9636bf87

[27] Oliver Korb, Thomas Stützle, and Thomas E. Exner. “PLANTS: Application of Ant Colony Optimization to Structure-Based Drug Design”. In: *Ant Colony Optimization and Swarm Intelligence*. Berlin, Germany: Springer, 2006, pp. 247–258. DOI: 10.1007/11839088_22.

[28] Irwin D. Kuntz et al. “A geometric approach to macromolecule-ligand interactions”. In: *J. Mol. Biol.* 161.2 (Oct. 1982), pp. 269–288. ISSN: 0022-2836. DOI: 10.1016/0022-2836(82)90153-X.

[29] Yujia Li et al. “Graph Matching Networks for Learning the Similarity of Graph Structured Objects”. In: *arXiv* (Apr. 2019). DOI: 10.48550/arXiv.1904.12787. eprint: 1904.12787.

[30] Libretexts. “Dissociation Constant”. In: *Chemistry LibreTexts* (June 2021). URL: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Equilibria/Chemical_Equilibria/Dissociation_Constant.

[31] C. A. Lipinski et al. “Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings”. In: *Adv. Drug Delivery Rev.* 46.1-3 (Mar. 2001), pp. 3–26. ISSN: 0169-409X. DOI: 10.1016/s0169-409x(00)00129-0. eprint: 11259830.

[32] Zhihai Liu et al. "Forging the Basis for Developing Protein–Ligand Interaction Scoring Functions". In: *Acc. Chem. Res.* 50.2 (Feb. 2017), pp. 302–309. ISSN: 0001-4842. DOI: 10.1021/acs.accounts.6b00491.

[33] Yu-Chen Lo et al. "Machine learning in chemoinformatics and drug discovery". In: *Drug Discovery Today* 23.8 (Aug. 2018), pp. 1538–1546. ISSN: 1359-6446. DOI: 10.1016/j.drudis.2018.05.010.

[34] Marc A. Moesser et al. "Protein-Ligand Interaction Graphs: Learning from Ligand-Shaped 3D Interaction Graphs to Improve Binding Affinity Prediction". In: *bioRxiv* (Mar. 2022), p. 2022.03.04.483012. eprint: 2022.03.04.483012. URL: https://doi.org/10.1101/2022.03.04.483012.

[35] H. L. Morgan. "The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service." In: *J. Chem. Doc.* 5.2 (May 1965), pp. 107–113. ISSN: 0021-9576. DOI: 10.1021/c160017a018.

[36] Garrett M. Morris and Marguerita Lim-Wilby. "Molecular Docking". In: *Molecular Modeling of Proteins*. Humana Press, 2008, pp. 365–382. DOI: 10.1007/978-1-59745-177-2_19.

[37] Garrett M. Morris et al. "AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility". In: *J. Comput. Chem.* 30.16 (Dec. 2009), p. 2785. DOI: 10.1002/jcc.21256.

[38] Michael M. Mysinger et al. "Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking". In: *J. Med. Chem.* 55.14 (July 2012), pp. 6582–6594. ISSN: 0022-2623. DOI: 10.1021/jm300687e.

[39] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *arXiv* (Nov. 2015). DOI: 10.48550/arXiv.1511.08458. eprint: 1511.08458.

[40] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *arXiv* (Dec. 2019). DOI: 10.48550/arXiv.1912.01703. eprint: 1912.01703.

[41] Steven M. Paul et al. "How to improve R&D productivity: the pharmaceutical industry's grand challenge". In: *Nat. Rev. Drug Discovery* 9 (Mar. 2010), pp. 203–214. ISSN: 1474-1784. DOI: 10.1038/nrd3078.

[42] Eric F. Pettersen et al. "UCSF Chimera–a visualization system for exploratory research and analysis". In: *J. Comput. Chem.* 25.13 (Oct. 2004), pp. 1605–1612. ISSN: 0192-8651. DOI: 10.1002/jcc.20084. eprint: 15264254.

[43] David Rogers and Mathew Hahn. "Extended-Connectivity Fingerprints". In: *J. Chem. Inf. Model.* 50.5 (May 2010), pp. 742–754. ISSN: 1549-9596. DOI: 10.1021/ci100050t.

[44] F. J. C. Rossotti and H. Rossotti. "The Determination of Stability Constants and Other Equilibrium Constants in Solution". In: *Series in Advanced Chemistry* 66.3 (Apr. 1962). ISSN: 0372-8382. DOI: 10.1002/bbpc.19620660326.

[45] Sergio Ruiz-Carmona et al. "rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids". In: *PLoS Comput. Biol.* 10.4 (Apr. 2014), e1003571. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003571.

[46] Norberto Sánchez-Cruz et al. "Extended connectivity interaction features: improving binding affinity prediction through chemical description". In: *Bioinformatics* 37.10 (May 2021), pp. 1376–1382. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btaa982.

[47] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. "E(n) Equivariant Graph Neural Networks". In: *arXiv* (Feb. 2021). DOI: 10.48550/arXiv.2102.09844. eprint: 2102.09844.

[48] Sakshi Singh, Qanita Bani Baker, and Dev Bukhsh Singh. "Molecular docking and molecular dynamics simulation". In: *Bioinformatics*. Cambridge, MA, USA: Academic Press, Jan. 2022, pp. 291–304. ISBN: 978-0-323-89775-4. DOI: 10.1016/B978-0-323-89775-4.00014-6.

[49] Christoph Sotriffer et al. *Virtual Screening: Principles, Challenges, and Practical Guidelines*. Weinheim, Germany: Wiley, Feb. 2011. ISBN: 978-3-527-32636-5. URL: https://www.wiley.com/en-us/Virtual+Screening%3A+Principles%2C+Challenges%2C+and+Practical+Guidelines-p-9783527326365.

[50] Hannes Stärk et al. "EquiBind: Geometric Deep Learning for Drug Binding Structure Prediction". In: *arXiv* (Feb. 2022). DOI: 10.48550/arXiv.2202.05146. eprint: 2202.05146.

[51] M. J. Stewart and I. D. Watson. "Standard units for expressing drug concentrations in biological fluids." In: *Br. J. Clin. Pharmacol.* 16.1 (July 1983), p. 3. DOI: 10.1111/j.1365-2125.1983.tb02136.x.

[52] Minyi Su et al. "Comparative Assessment of Scoring Functions: The CASF-2016 Update". In: *J. Chem. Inf. Model.* 59.2 (Feb. 2019), pp. 895–913. ISSN: 1549-960X. DOI: 10.1021/acs.jcim.8b00545. eprint: 30481020.

[53] MDL Information Systems. *CTFile Formats (PDF)*. June 2005. URL: https://web.archive.org/web/20070630061308/http://www.mdl.com/downloads/public/ctfile/ctfile.pdf.

[54] Petar Veličković et al. "Graph Attention Networks". In: *arXiv* (Oct. 2017). DOI: 10.48550/arXiv.1710.10903. eprint: 1710.10903.

[55] Santiago Vilar, Giorgio Cozza, and Stefano Moro. "Medicinal chemistry and the molecular operating environment (MOE): application of QSAR and molecular docking to drug discovery". In: *Curr. Top. Med. Chem.* 8.18 (2008), pp. 1555–1572. ISSN: 1873-4294. DOI: 10.2174/156802608786786624. eprint: 19075767.

[56] Junmei Wang et al. "Development and testing of a general amber force field". In: *J. Comput. Chem.* 25.9 (July 2004), pp. 1157–1174. ISSN: 0192-8651. DOI: 10.1002/jcc.20035. eprint: 15116359.

[57] Minjie Wang et al. "Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks". In: *arXiv* (Sept. 2019). DOI: 10.48550/arXiv.1909.01315. eprint: 1909.01315.

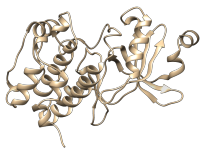
[58] David Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *J. Chem. Inf. Comput. Sci.* 28.1 (Feb. 1988), pp. 31–36. ISSN: 0095-2338. DOI: 10.1021/ci00057a005.

[59] Gengmo Zhou et al. "Uni-Mol: A Universal 3D Molecular Representation Learning Framework". In: *ChemRxiv* (Sept. 2022). DOI: 10.26434/chemrxiv-2022-jjm0j-v3.

[60] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (Jan. 2020), pp. 57–81. ISSN: 2666-6510. DOI: 10.1016/j.aiopen.2021.01.001.

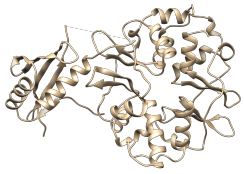
Appendix A

Redocking Targets

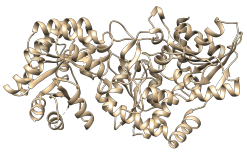


(a) Calmodulin-domain protein kinase 1

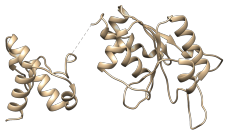
(b) PA-I galactophilic lectin



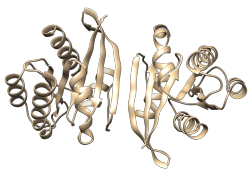
(c) 5'-AMP-activated protein kinase cat- alytic subunit A



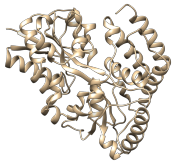
(d) Tryptophan synthase alpha chain



(e) tRNA (guanine-N1-)-methyltransferase



(f) GTPase KRas



(g) ABC transporter, periplasmic substrate-binding protein

Figure A.1: Caption

Appendix B

Redocking Crystal Poses - Further Results

Training set

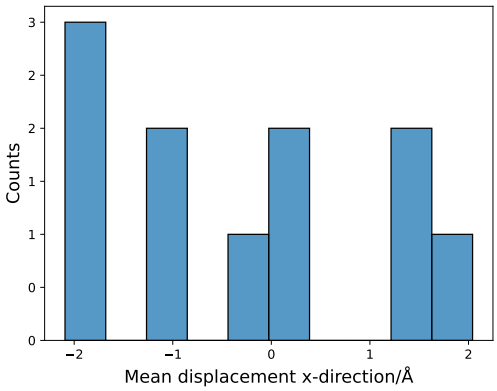
PA-I galactophilic lectin

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-0.282	1.265	0.451	1.854	6.198	1.181
std	1.522	5.533	1.301	0.588	3.290	1.701
min	-2.093	-6.332	-0.405	0.899	1.352	0.157
25%	-1.549	-1.950	-0.263	1.437	3.953	0.385
50%	-0.238	-0.014	-0.007	2.156	6.717	0.442
75%	0.889	4.988	0.432	2.233	8.534	0.713
max	2.041	10.502	3.950	2.622	11.827	5.475

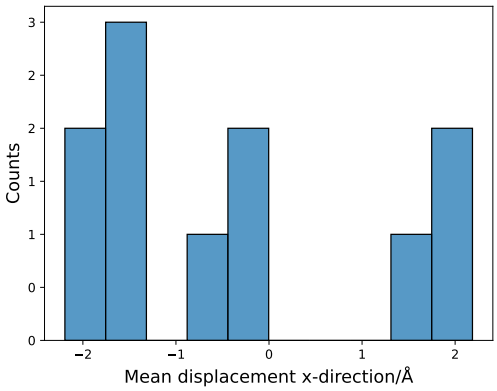
Table B.1: PA-I galactophilic lectin docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-0.359	0.568	11.056	1.421	4.470	10.465
std	1.553	5.798	4.961	0.619	2.884	4.672
min	-2.193	-9.096	2.863	0.150	0.753	2.713
25%	-1.522	-2.710	7.707	1.162	2.432	7.299
50%	-0.655	-0.386	11.705	1.505	4.089	11.052
75%	0.680	4.903	14.189	1.894	6.346	13.402
max	2.186	9.726	19.655	2.072	9.170	18.576

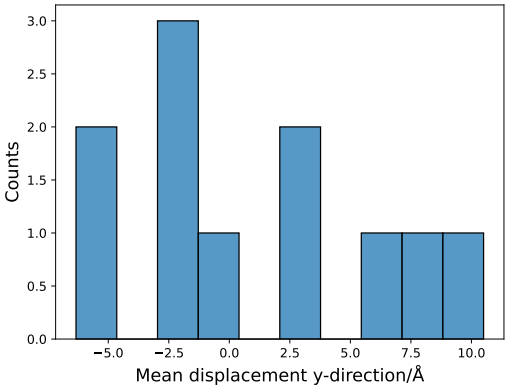
Table B.2: PA-I galactophilic lectin redocked metrics/Å



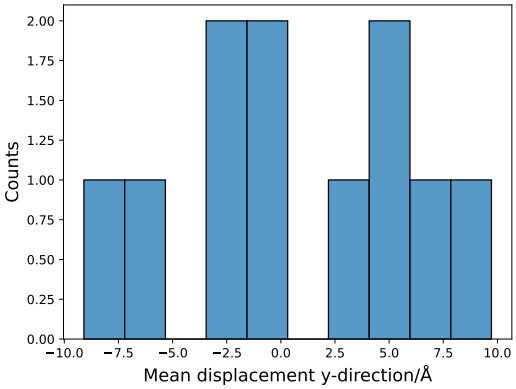
(a) Docked



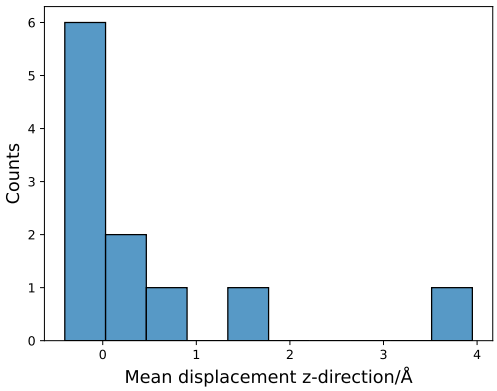
(b) Redocked



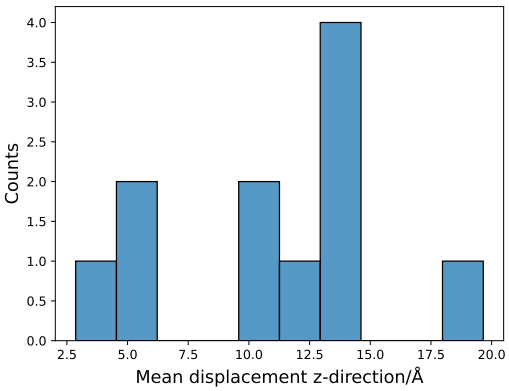
(c) Docked



(d) Redocked

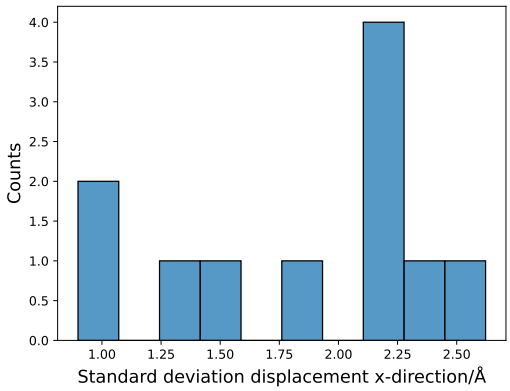


(e) Docked

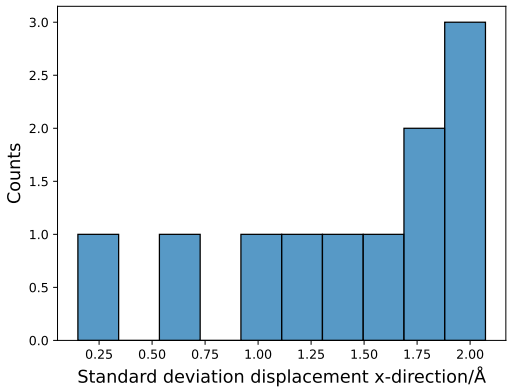


(f) Redocked

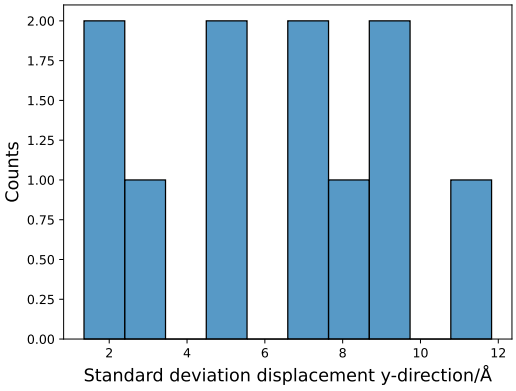
Figure B.1: PA-I galactophilic lectin mean displacements



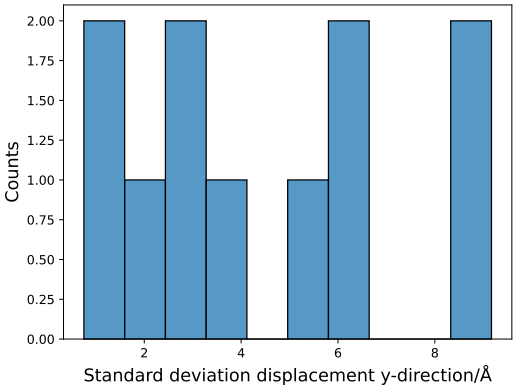
(a) Docked



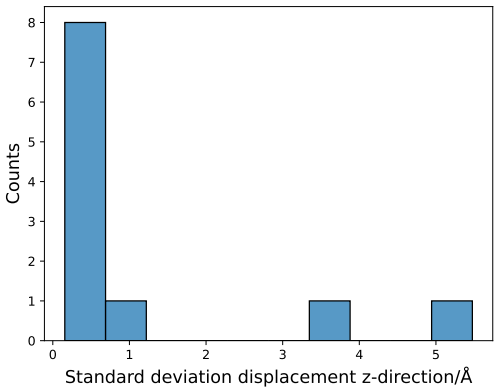
(b) Redocked



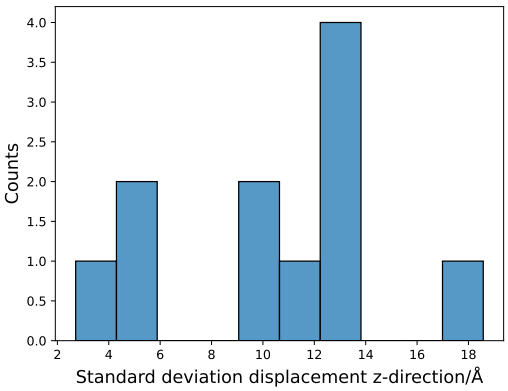
(c) Docked



(d) Redocked



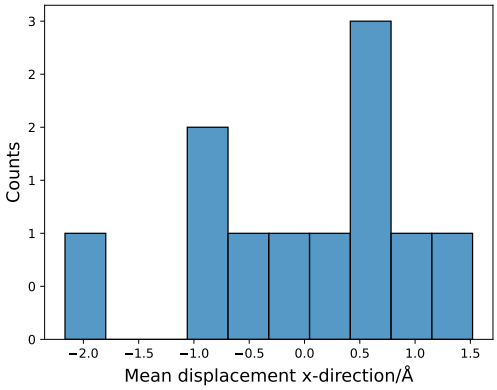
(e) Docked



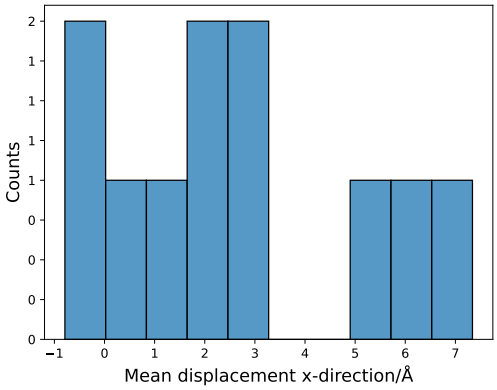
(f) Redocked

Figure B.2: PA-I galactophilic lectin standard deviation displacements

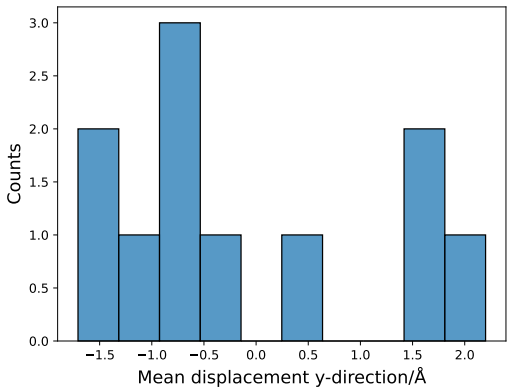
5'-AMP-activated protein kinase catalytic subunit A



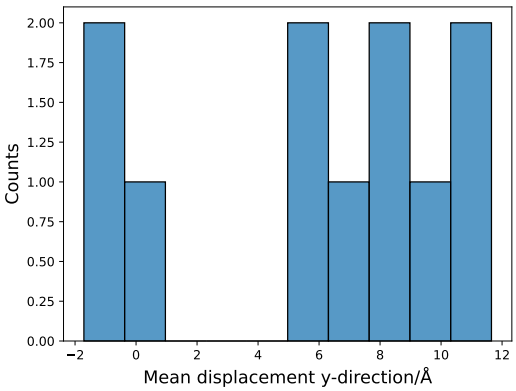
(a) Docked



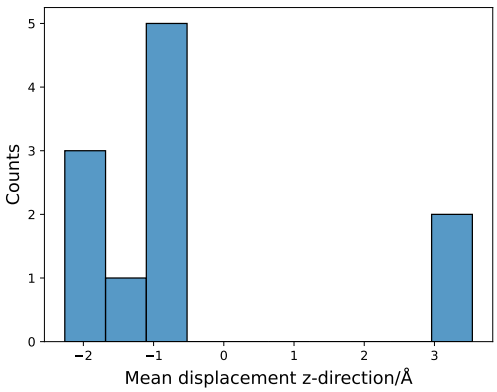
(b) Redocked



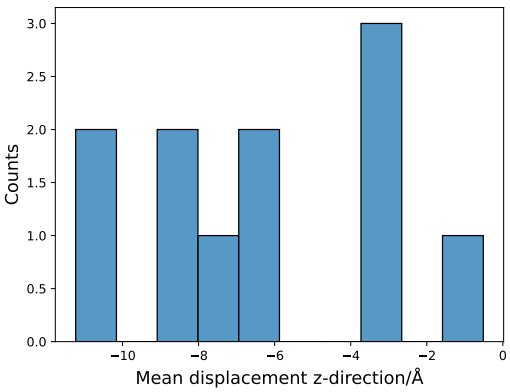
(c) Docked



(d) Redocked

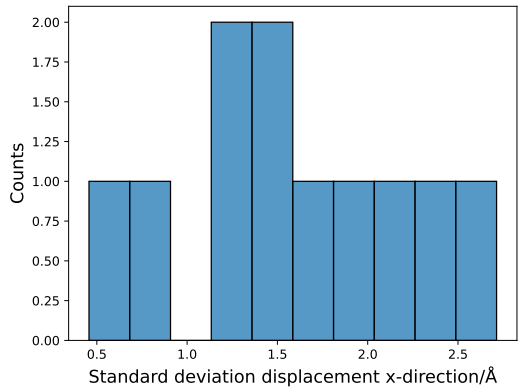


(e) Docked

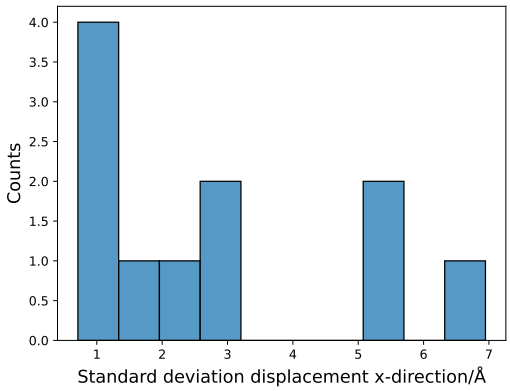


(f) Redocked

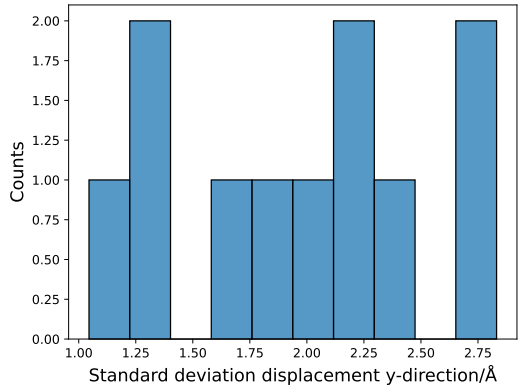
Figure B.3: 5'-AMP-activated protein kinase catalytic subunit A mean displacements



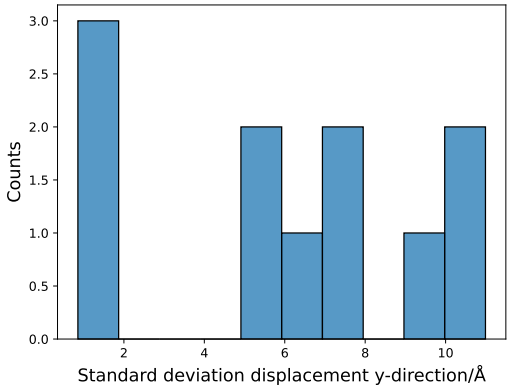
(a) Docked



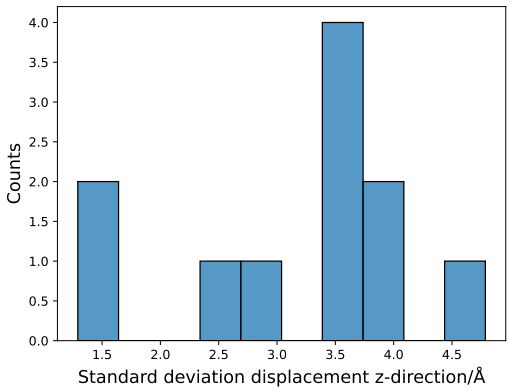
(b) Redocked



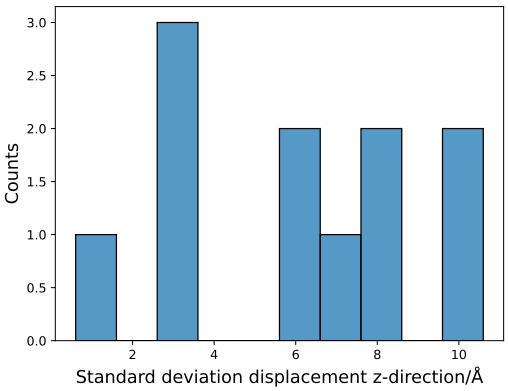
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure B.4: 5'-AMP-activated protein kinase catalytic subunit A standard deviation displacements

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-0.035	-0.126	-0.487	1.583	1.944	3.155
std	1.059	1.367	1.978	0.674	0.585	1.069
min	-2.166	-1.707	-2.264	0.457	1.045	1.292
25%	-0.723	-1.065	-1.710	1.251	1.501	2.692
50%	0.276	-0.742	-0.970	1.474	1.944	3.467
75%	0.575	1.006	-0.701	1.982	2.297	3.746
max	1.520	2.200	3.540	2.714	2.832	4.786

Table B.3: 5'-AMP-activated protein kinase catalytic subunit A docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	2.663	5.868	-6.238	2.791	6.130	5.946
std	2.667	4.721	3.422	2.188	3.553	3.168
min	-0.790	-1.710	-11.231	0.711	0.856	0.605
25%	1.078	3.087	-8.530	1.066	3.629	3.186
50%	2.418	6.423	-6.886	2.289	6.125	6.515
75%	4.209	8.932	-3.206	3.973	8.431	8.055
max	7.341	11.652	-0.515	6.947	10.991	10.598

Table B.4: 5'-AMP-activated protein kinase catalytic subunit A redocked metrics/Å

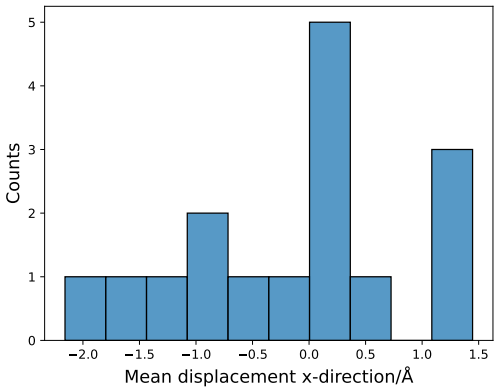
Tryptophan synthase alpha chain

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-0.124	0.129	0.543	1.263	1.774	1.361
std	1.018	1.949	0.898	0.418	1.349	0.521
min	-2.159	-2.558	-0.864	0.594	0.527	0.576
25%	-0.843	-0.658	-0.319	0.964	0.921	0.902
50%	0.144	-0.212	0.689	1.215	1.453	1.399
75%	0.443	0.446	1.206	1.530	2.142	1.789
max	1.446	5.939	2.011	2.256	5.971	2.128

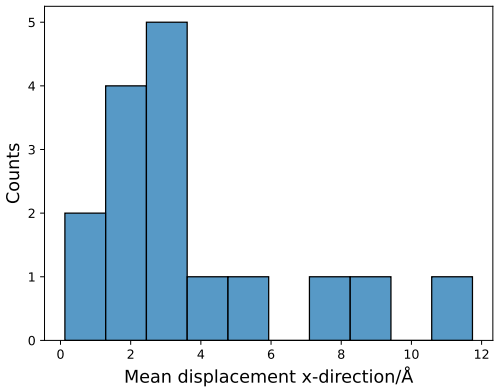
Table B.5: Tryptophan synthase alpha chain docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	3.767	-0.909	0.893	3.656	1.659	2.277
std	3.157	2.067	2.782	2.942	1.314	1.476
min	0.125	-4.002	-3.115	0.414	0.287	0.343
25%	2.033	-2.290	-1.538	1.981	0.568	1.298
50%	3.093	-0.717	0.576	2.973	1.009	1.856
75%	4.107	-0.370	3.173	3.981	2.635	2.996
max	11.740	4.220	5.925	11.159	3.980	5.587

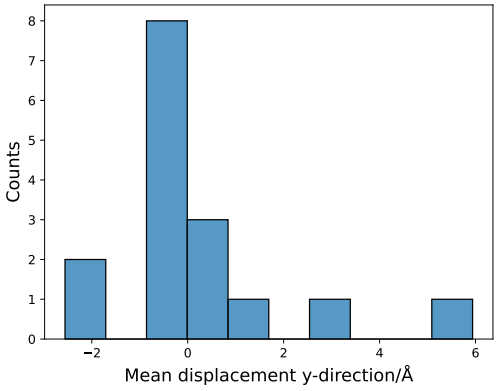
Table B.6: Tryptophan synthase alpha chain redocked metrics/Å



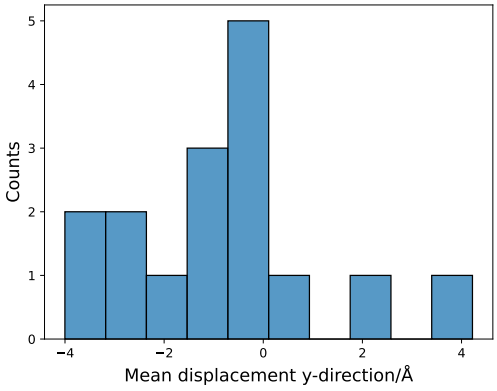
(a) Docked



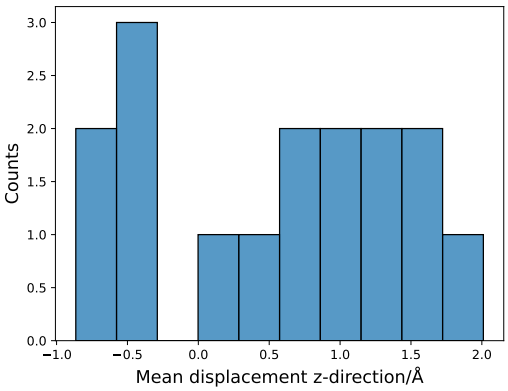
(b) Redocked



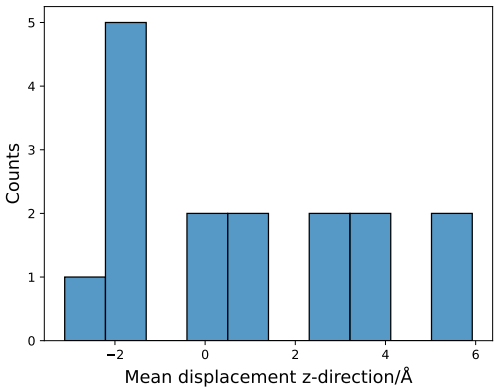
(c) Docked



(d) Redocked

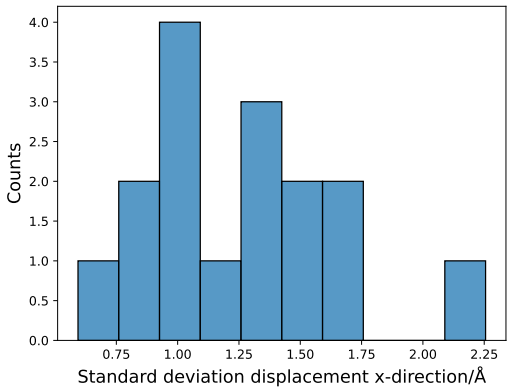


(e) Docked

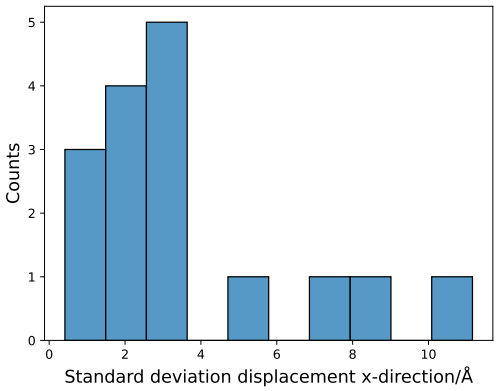


(f) Redocked

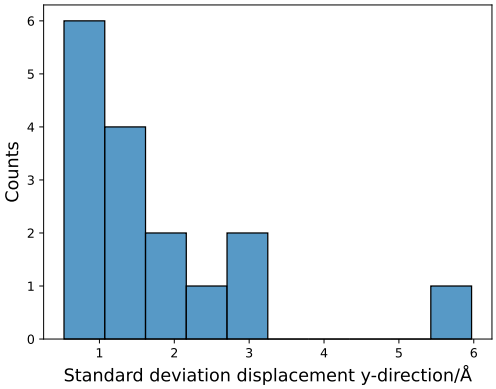
Figure B.5: Tryptophan synthase alpha chain mean displacements



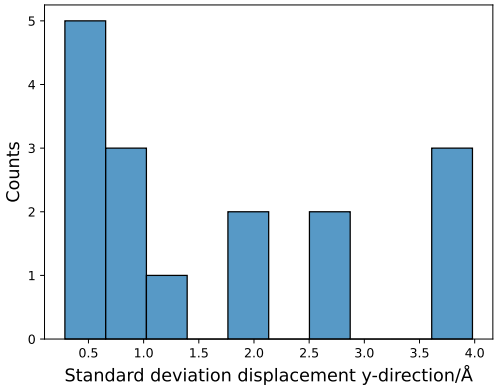
(a) Docked



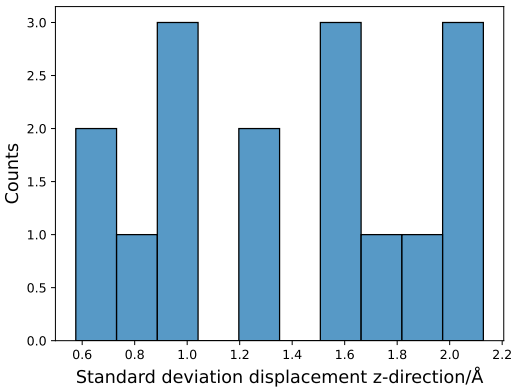
(b) Redocked



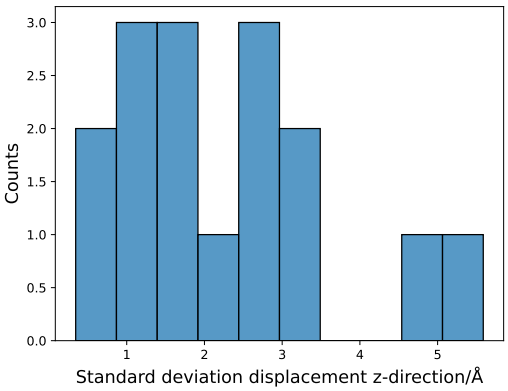
(c) Docked



(d) Redocked



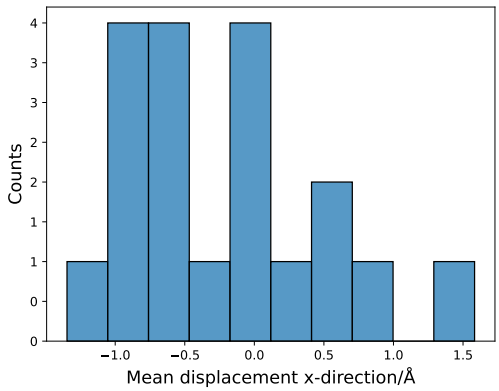
(e) Docked



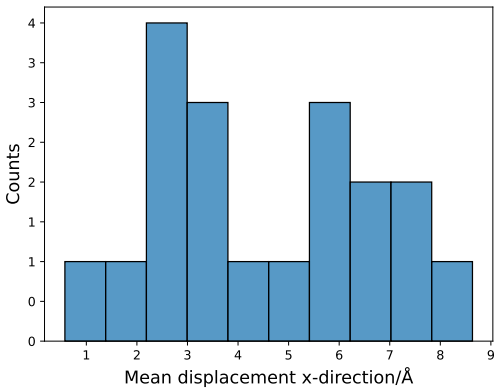
(f) Redocked

Figure B.6: Tryptophan synthase alpha chain standard deviation displacements

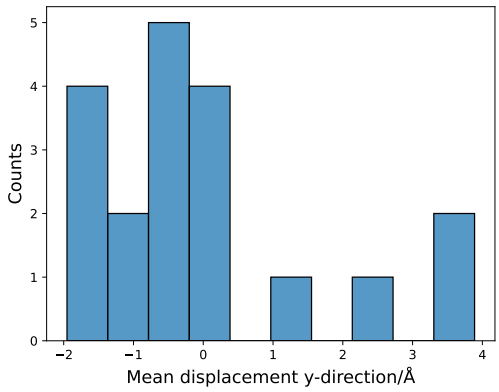
tRNA (guanine-N1-)-methyltransferase



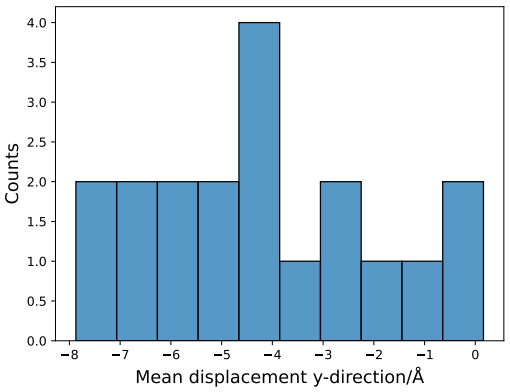
(a) Docked



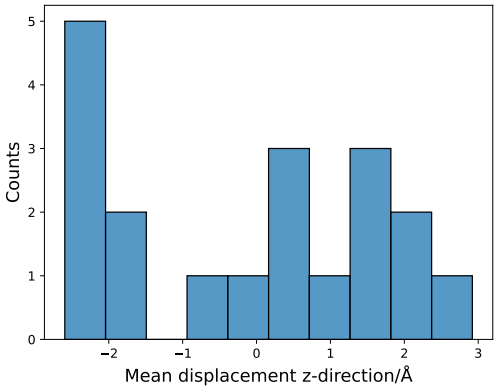
(b) Redocked



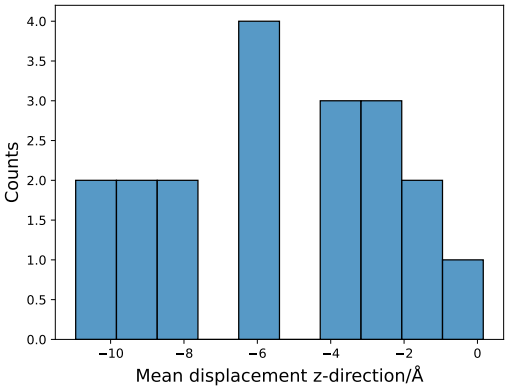
(c) Docked



(d) Redocked

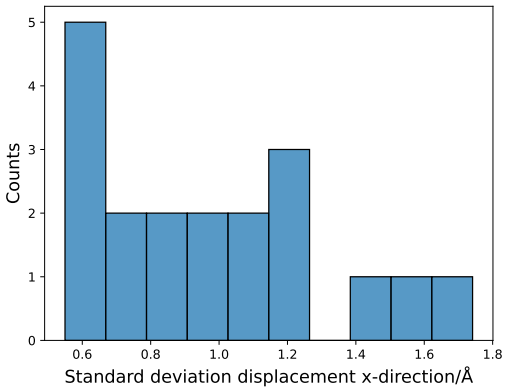


(e) Docked

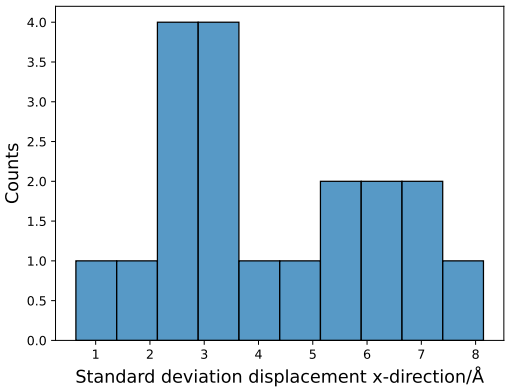


(f) Redocked

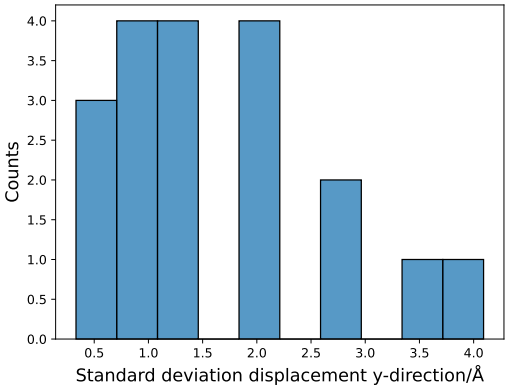
Figure B.7: tRNA (guanine-N1-)-methyltransferase mean displacements



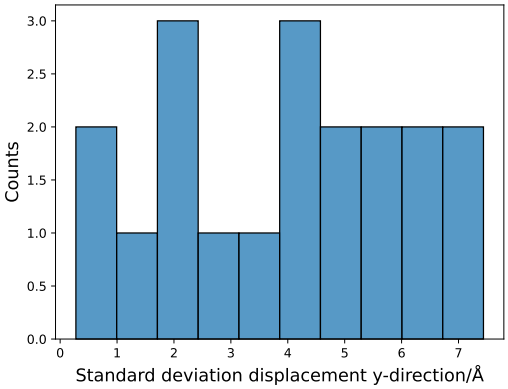
(a) Docked



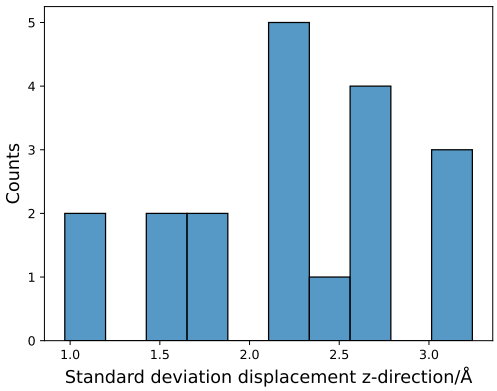
(b) Redocked



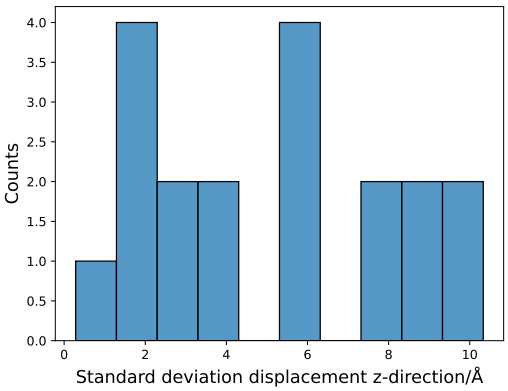
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure B.8: tRNA (guanine-N1-)-methyltransferase standard deviation displacements

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-0.243	-0.004	-0.094	0.984	1.662	2.211
std	0.740	1.686	1.820	0.330	1.061	0.654
min	-1.346	-1.953	-2.595	0.549	0.332	0.970
25%	-0.811	-0.933	-1.906	0.720	0.896	1.842
50%	-0.461	-0.547	0.186	0.948	1.406	2.214
75%	0.157	0.161	1.479	1.155	2.112	2.654
max	1.583	3.889	2.921	1.741	4.092	3.242

Table B.7: tRNA (guanine-N1-)-methyltransferase docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	4.561	-4.240	-5.373	4.323	4.038	5.108
std	2.268	2.407	3.311	2.124	2.220	3.074
min	0.576	-7.872	-10.952	0.636	0.277	0.285
25%	2.867	-6.036	-8.460	2.720	2.297	2.500
50%	3.816	-4.507	-5.723	3.605	4.259	5.404
75%	6.355	-2.425	-2.625	6.012	5.708	7.988
max	8.636	0.161	0.158	8.145	7.439	10.333

Table B.8: tRNA (guanine-N1-)-methyltransferase redocked metrics/Å

Test set

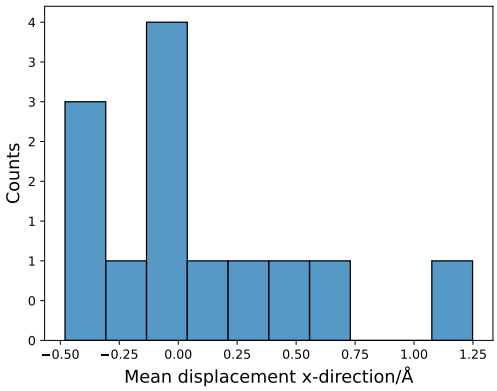
ABC transporter, periplasmic substrate-binding protein

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.085	-0.140	0.067	0.556	0.926	0.605
std	0.486	0.848	0.458	0.270	0.430	0.197
min	-0.480	-1.388	-0.501	0.295	0.332	0.343
25%	-0.156	-0.739	-0.412	0.393	0.623	0.519
50%	-0.069	-0.380	0.163	0.539	0.905	0.551
75%	0.349	0.345	0.370	0.602	1.106	0.616
max	1.248	1.707	0.904	1.329	1.726	1.074

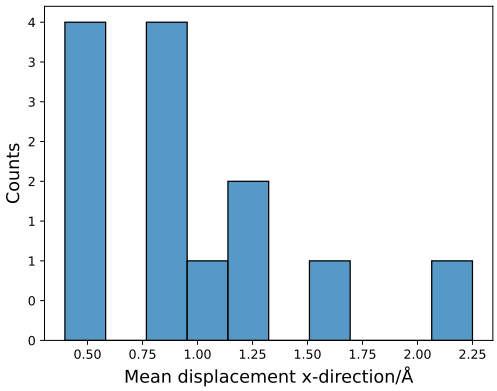
Table B.9: ABC transporter, periplasmic substrate-binding protein docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.972	2.801	-0.649	0.963	2.658	0.814
std	0.526	1.058	0.621	0.477	0.989	0.411
min	0.397	1.084	-1.564	0.429	1.069	0.248
25%	0.507	2.292	-0.992	0.574	2.225	0.502
50%	0.928	2.592	-0.822	0.904	2.449	0.888
75%	1.142	3.314	-0.220	1.108	3.156	0.995
max	2.250	5.272	0.491	2.144	4.971	1.524

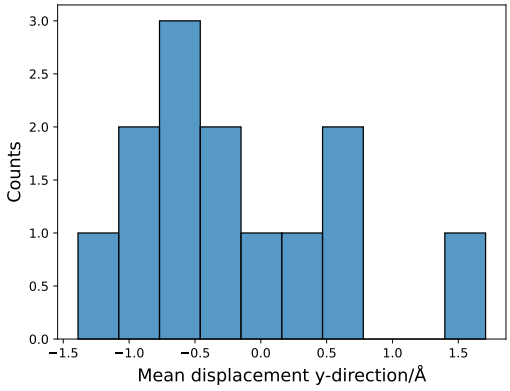
Table B.10: ABC transporter, periplasmic substrate-binding protein redocked metrics/Å



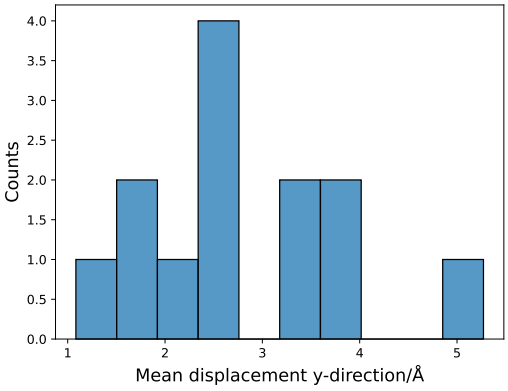
(a) Docked



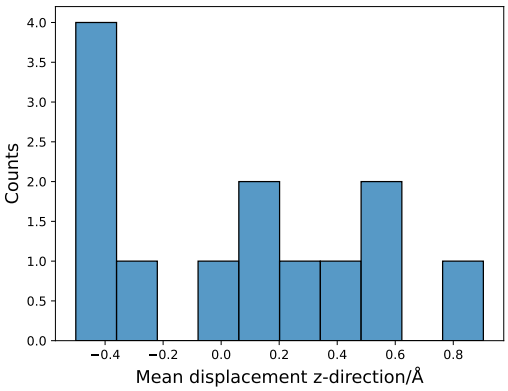
(b) Redocked



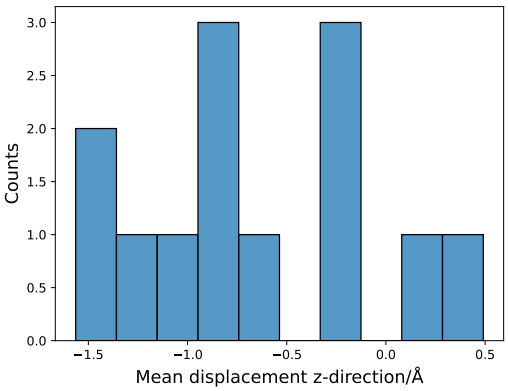
(c) Docked



(d) Redocked

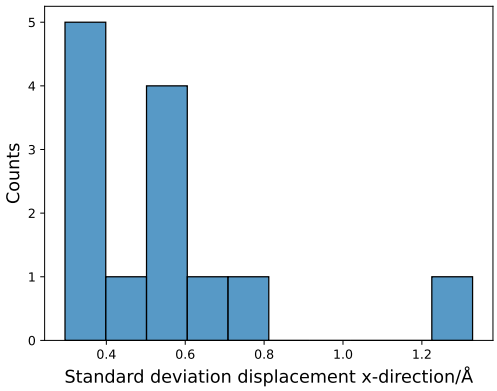


(e) Docked

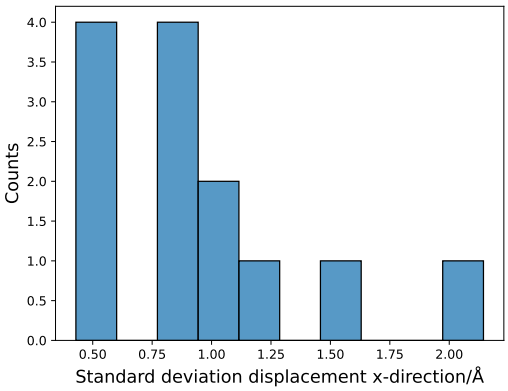


(f) Redocked

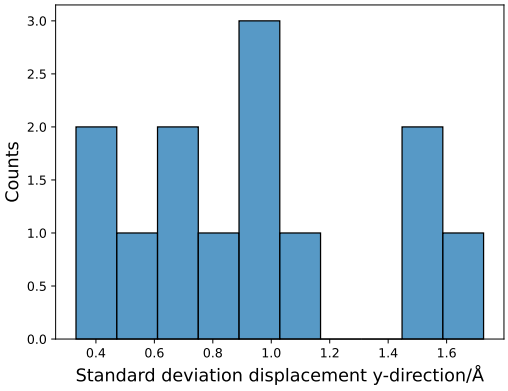
Figure B.9: ABC transporter, periplasmic substrate-binding protein mean displacements



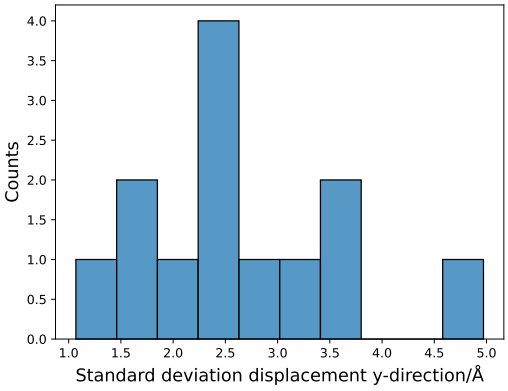
(a) Docked



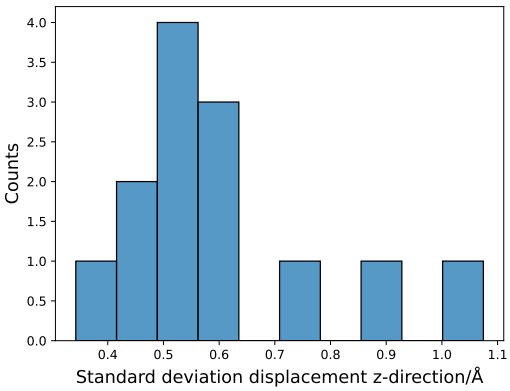
(b) Redocked



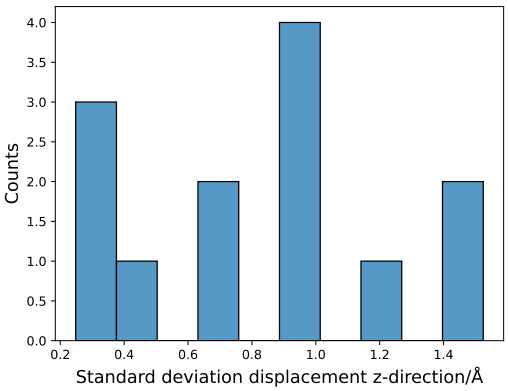
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure B.10: ABC transporter, periplasmic substrate-binding protein standard deviation displacements

Appendix C

Redocking Generated Conformers - Further Results

Training set

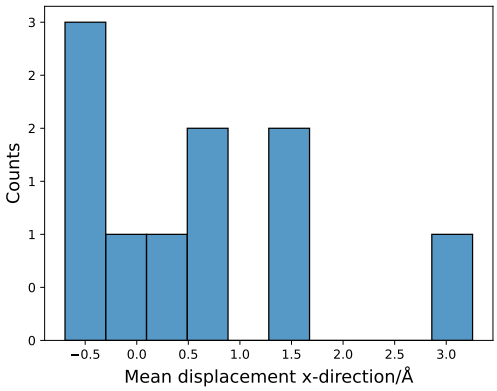
PA-I galactophilic lectin

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.703	1.341	0.475	2.272	4.632	1.637
std	1.382	3.388	0.852	1.057	2.629	1.432
min	-0.795	-6.671	-0.432	0.949	1.120	0.236
25%	-0.397	0.404	-0.140	1.472	2.386	0.530
50%	0.492	1.314	0.226	1.918	5.105	0.932
75%	1.420	3.423	0.860	3.318	6.404	2.818
max	3.720	5.593	2.027	3.770	8.191	4.084

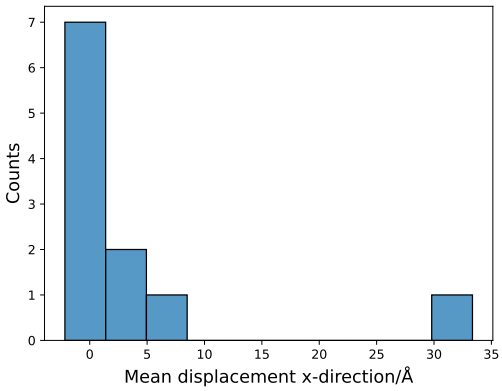
Table C.1: PA-I galactophilic lectin docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	3.700	6.337	17.700	4.114	7.223	16.294
std	10.031	16.138	27.232	8.626	13.703	24.244
min	-2.164	-5.827	2.434	0.496	0.275	2.301
25%	-0.628	0.191	5.405	0.653	1.864	5.136
50%	0.603	2.648	12.990	1.226	2.577	12.312
75%	2.108	5.431	13.552	2.332	5.514	12.835
max	33.351	53.743	98.726	29.831	48.070	88.310

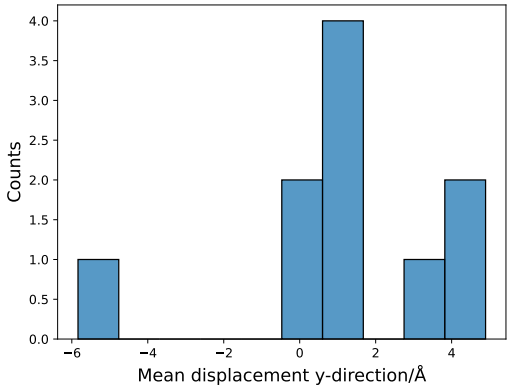
Table C.2: PA-I galactophilic lectin redocked metrics/Å



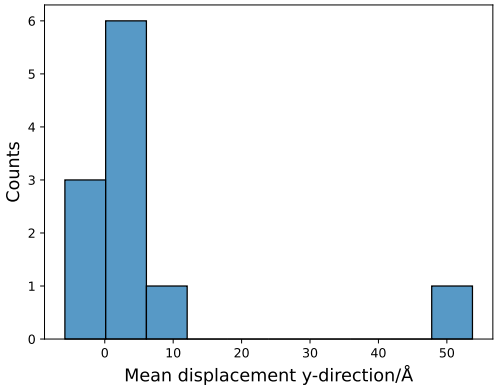
(a) Docked



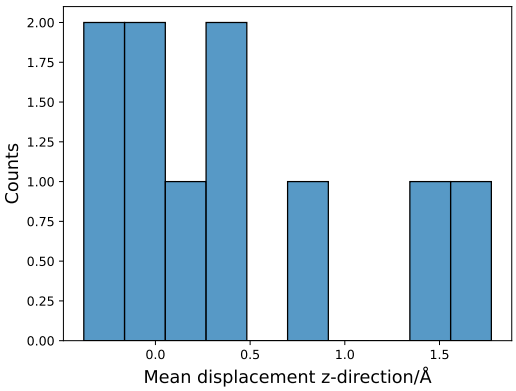
(b) Redocked



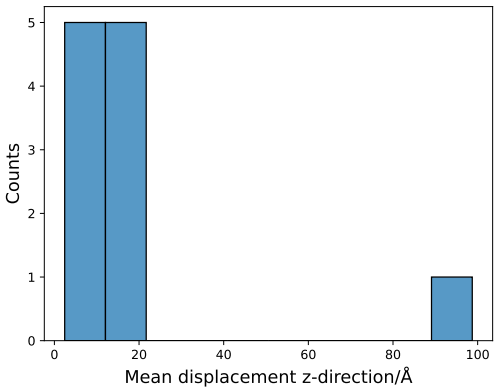
(c) Docked



(d) Redocked

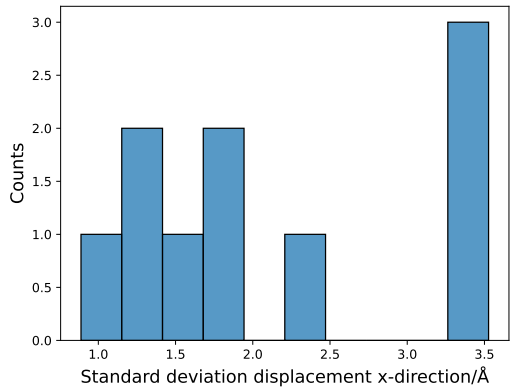


(e) Docked

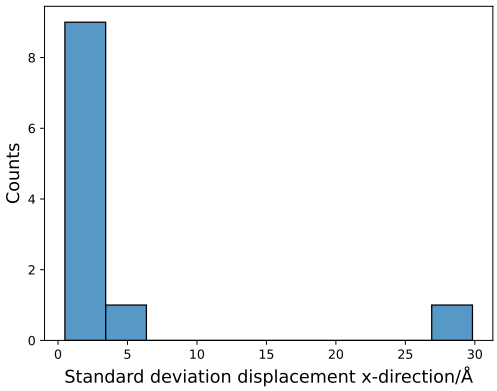


(f) Redocked

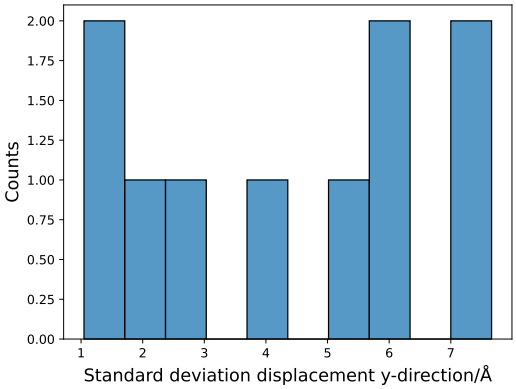
Figure C.1: PA-I galactophilic lectin mean displacements



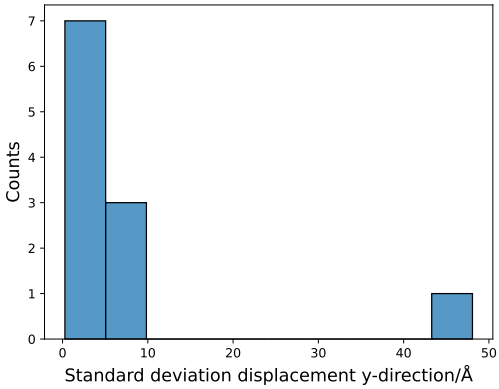
(a) Docked



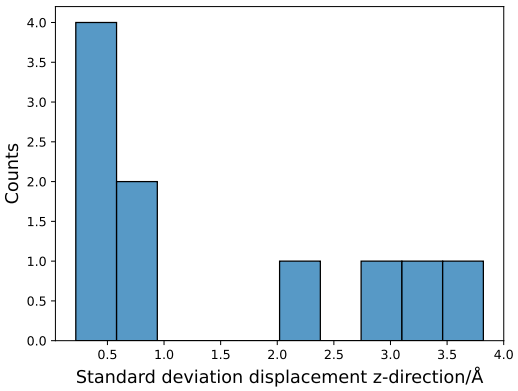
(b) Redocked



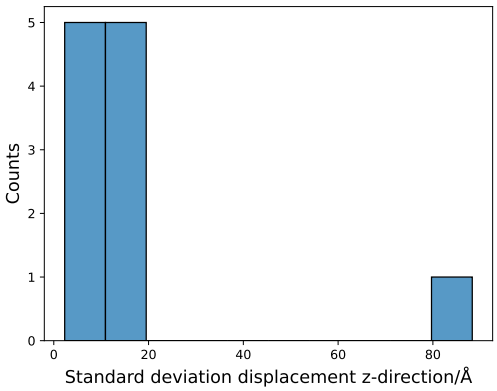
(c) Docked



(d) Redocked



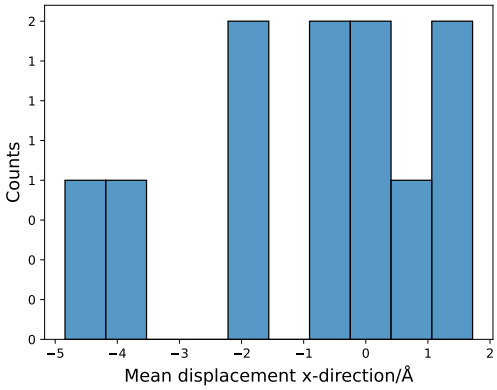
(e) Docked



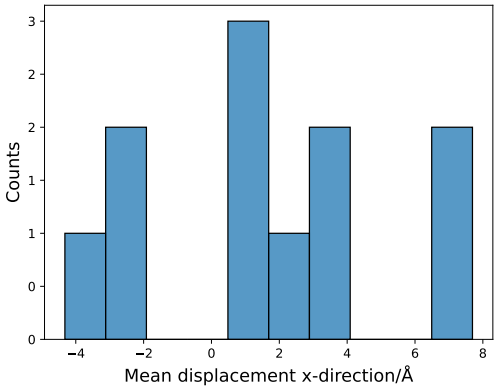
(f) Redocked

Figure C.2: PA-I galactophilic lectin standard deviation displacements

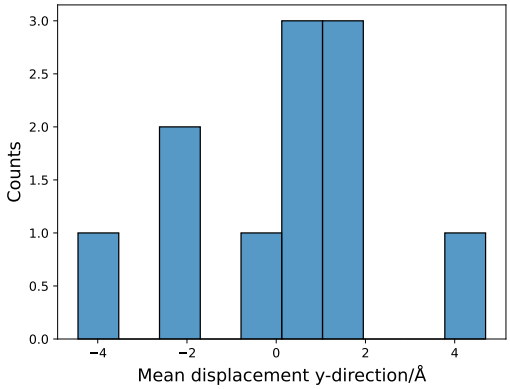
5'-AMP-activated protein kinase catalytic subunit A



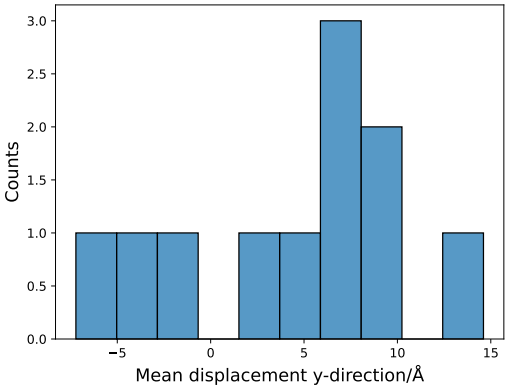
(a) Docked



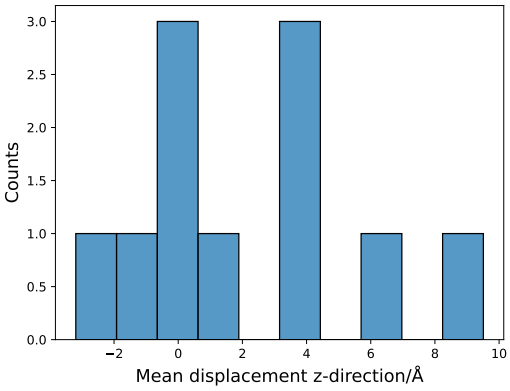
(b) Redocked



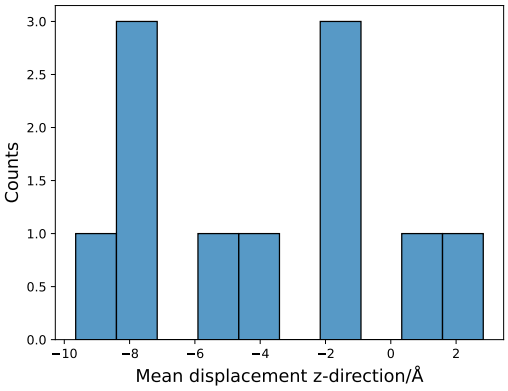
(c) Docked



(d) Redocked

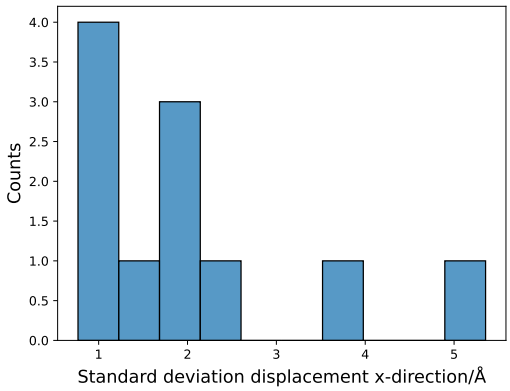


(e) Docked

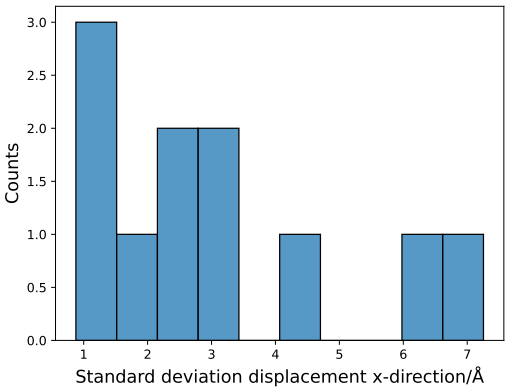


(f) Redocked

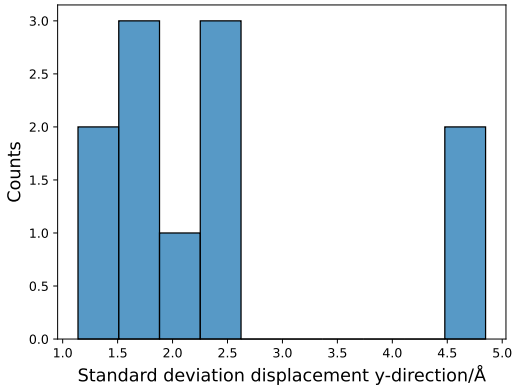
Figure C.3: 5'-AMP-activated protein kinase catalytic subunit A mean displacements



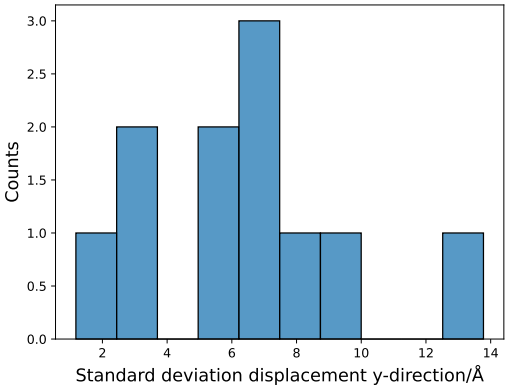
(a) Docked



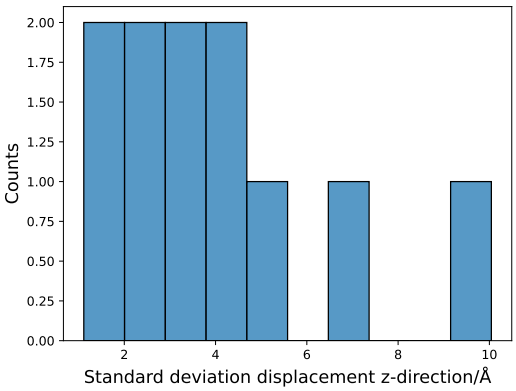
(b) Redocked



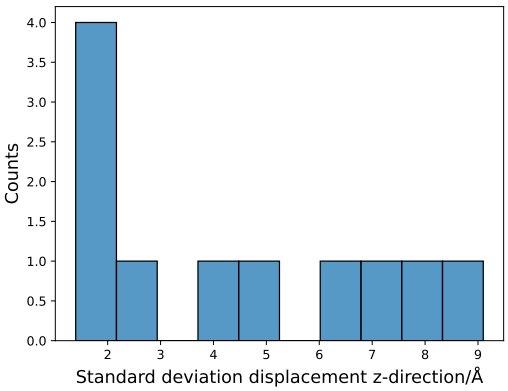
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure C.4: 5'-AMP-activated protein kinase catalytic subunit A standard deviation displacements

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-0.777	0.215	2.424	2.057	2.346	4.142
std	2.113	2.414	3.643	1.408	1.263	2.535
min	-4.842	-4.442	-3.192	0.767	1.140	1.113
25%	-1.878	-1.246	0.245	1.054	1.536	2.753
50%	-0.259	0.546	1.538	1.882	1.959	3.704
75%	0.425	1.558	4.169	2.146	2.334	4.536
max	1.719	4.692	9.510	5.354	4.849	10.044

Table C.3: 5'-AMP-activated protein kinase catalytic subunit A docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	1.554	4.702	-3.911	3.098	6.389	4.408
std	3.742	6.313	4.047	2.074	3.529	2.779
min	-4.321	-7.216	-9.651	0.878	1.180	1.395
25%	-0.785	0.752	-7.388	1.517	4.087	1.778
50%	1.389	6.432	-3.875	2.542	6.452	3.732
75%	3.532	8.516	-1.767	3.691	8.048	6.800
max	7.694	14.608	2.832	7.254	13.777	9.100

Table C.4: 5'-AMP-activated protein kinase catalytic subunit A redocked metrics/Å

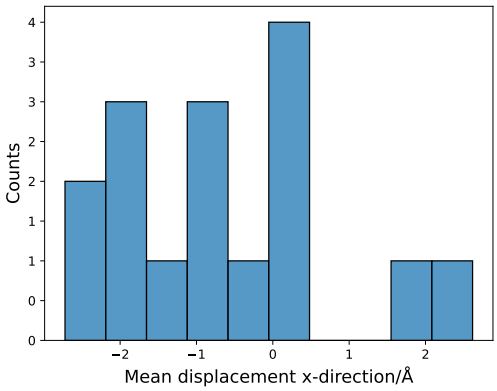
Tryptophan synthase alpha chain

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-0.597	0.613	0.397	1.948	2.273	1.573
std	1.502	2.133	1.439	0.687	1.349	0.597
min	-2.722	-3.994	-2.052	0.517	0.712	0.605
25%	-1.772	-0.363	-0.794	1.505	1.669	1.226
50%	-0.709	0.068	0.519	2.020	1.937	1.530
75%	0.219	1.479	1.586	2.495	2.156	2.172
max	2.617	5.411	2.335	2.844	5.717	2.466

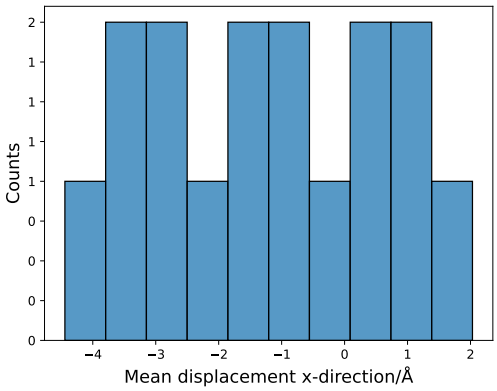
Table C.5: Tryptophan synthase alpha chain docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-1.165	0.020	2.229	1.824	1.468	2.326
std	1.908	2.081	2.124	1.106	1.273	1.739
min	-4.443	-5.058	-0.817	0.619	0.265	0.494
25%	-2.564	-0.850	0.594	0.956	0.721	0.822
50%	-1.021	-0.453	2.148	1.363	1.146	2.053
75%	0.450	1.475	3.334	2.446	1.556	3.155
max	2.031	4.423	6.278	4.202	4.775	5.922

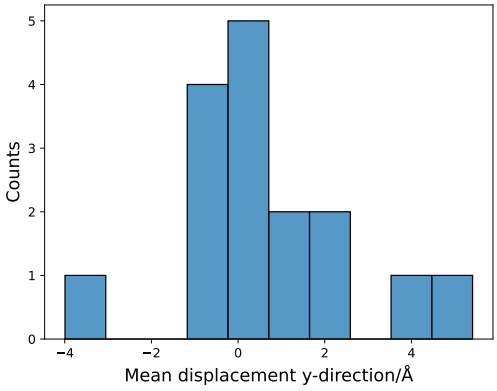
Table C.6: Tryptophan synthase alpha chain redocked metrics/Å



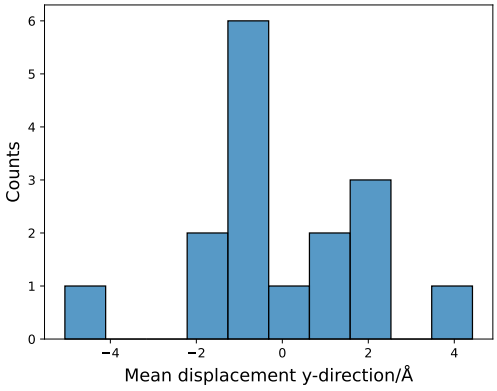
(a) Docked



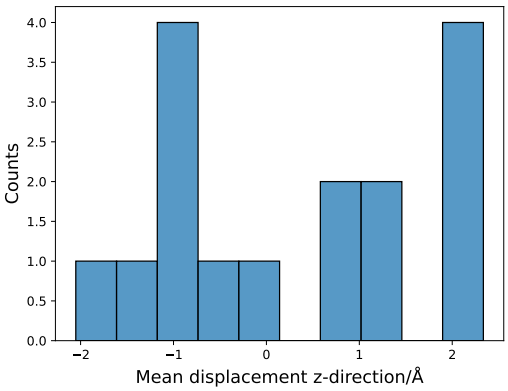
(b) Redocked



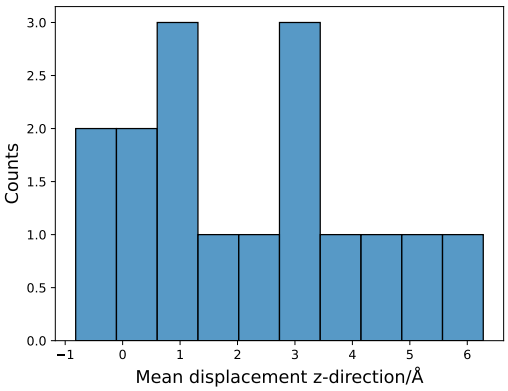
(c) Docked



(d) Redocked

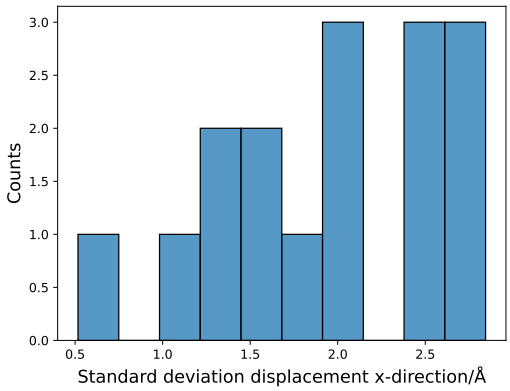


(e) Docked

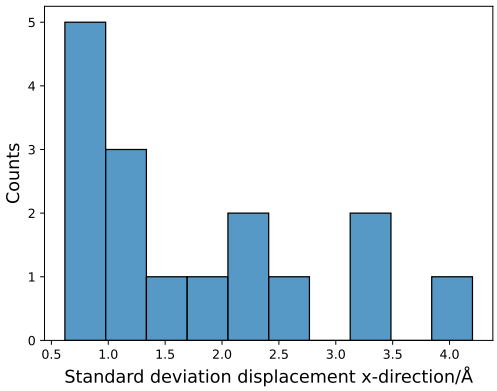


(f) Redocked

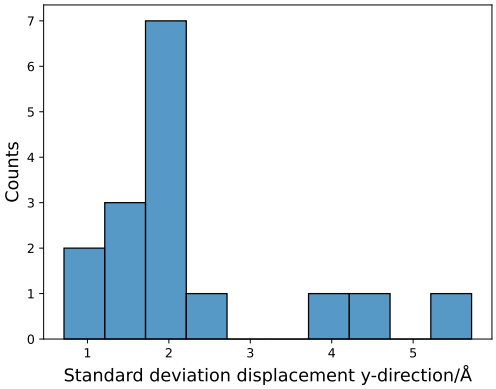
Figure C.5: Tryptophan synthase alpha chain mean displacements



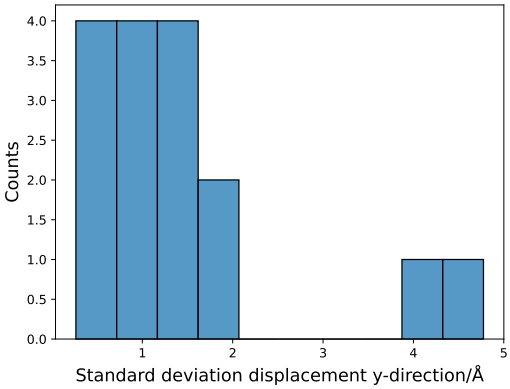
(a) Docked



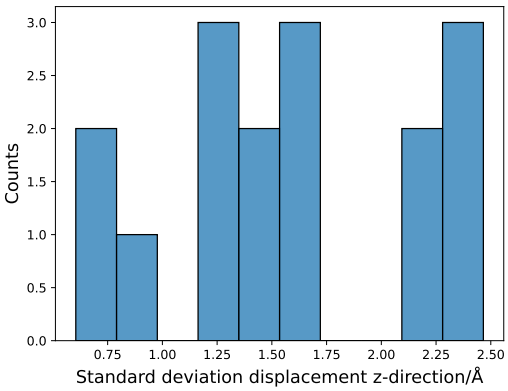
(b) Redocked



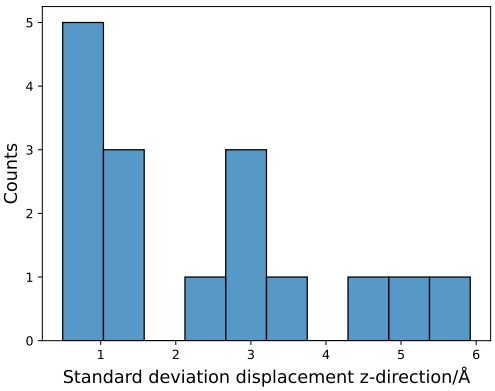
(c) Docked



(d) Redocked



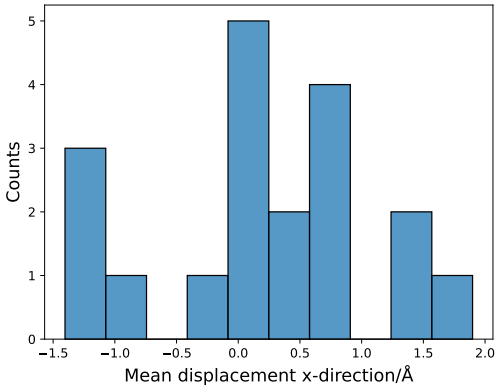
(e) Docked



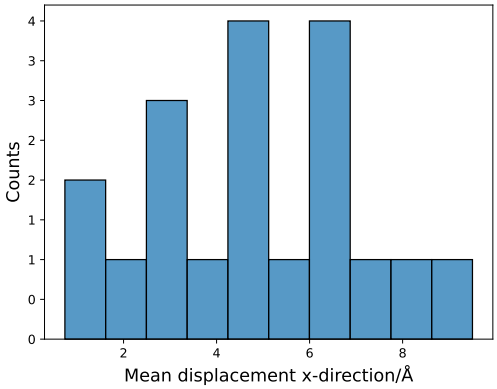
(f) Redocked

Figure C.6: Tryptophan synthase alpha chain standard deviation displacements

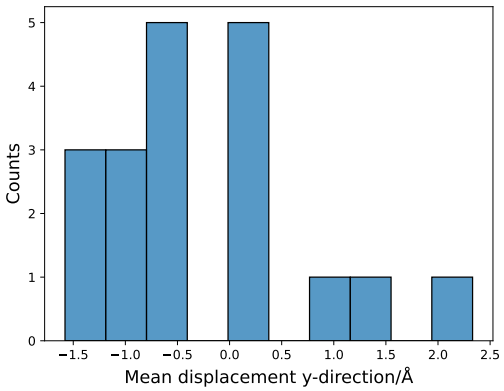
tRNA (guanine-N1-)-methyltransferase



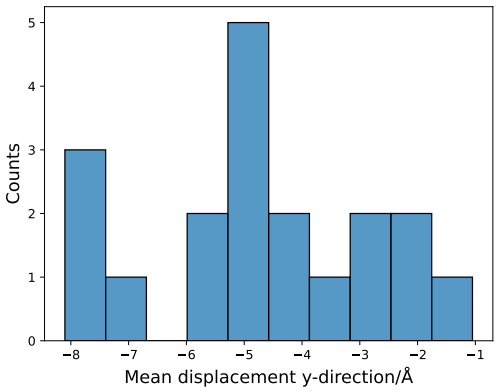
(a) Docked



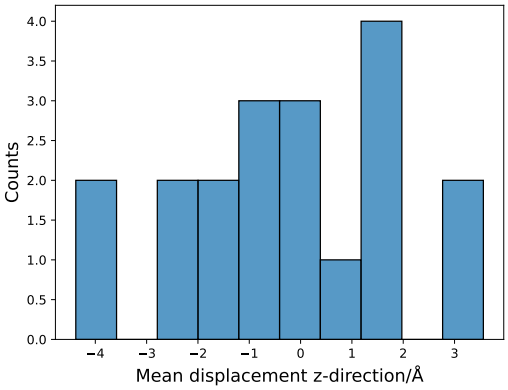
(b) Redocked



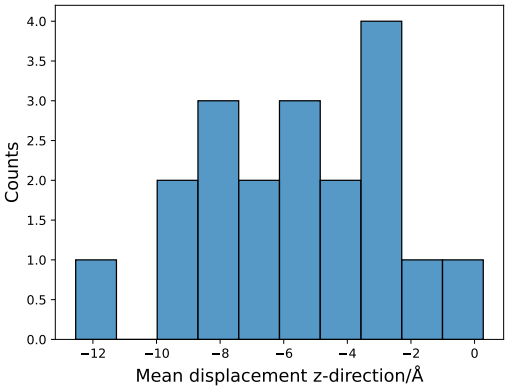
(c) Docked



(d) Redocked

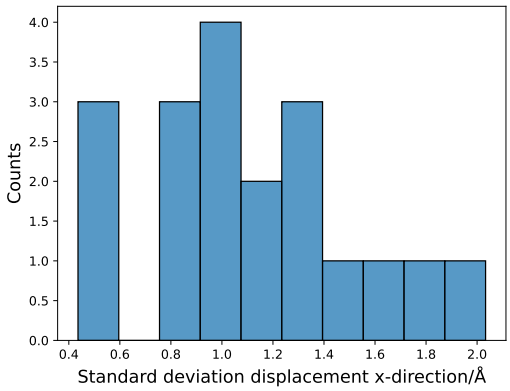


(e) Docked

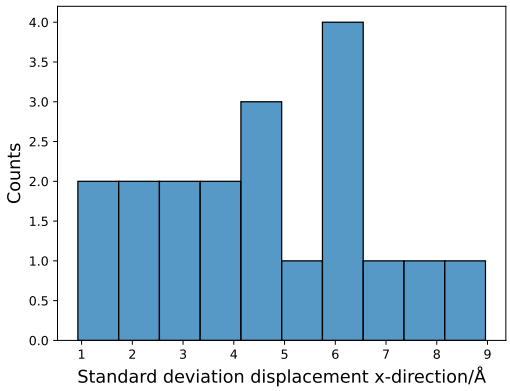


(f) Redocked

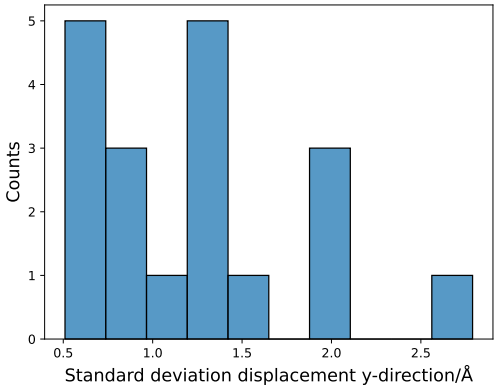
Figure C.7: tRNA (guanine-N1-)-methyltransferase mean displacements



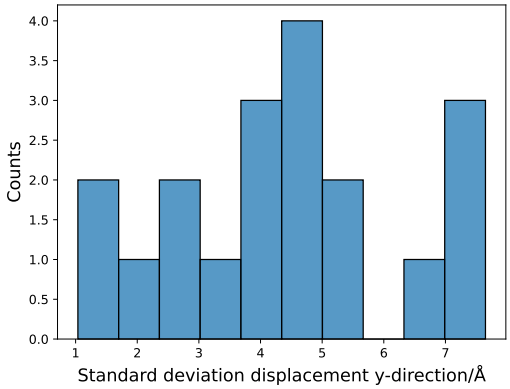
(a) Docked



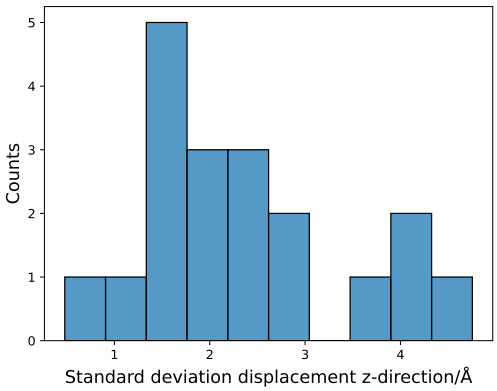
(b) Redocked



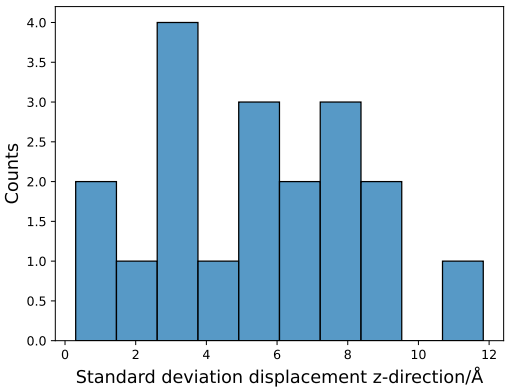
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure C.8: tRNA (guanine-N1-)-methyltransferase standard deviation displacements

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.201	-0.197	-0.271	1.124	1.236	2.353
std	0.907	0.984	2.174	0.419	0.622	1.160
min	-1.404	-1.579	-4.380	0.436	0.510	0.480
25%	-0.205	-0.855	-1.375	0.888	0.734	1.505
50%	0.173	-0.415	-0.174	1.070	1.215	1.994
75%	0.733	0.217	1.264	1.370	1.446	2.945
max	1.899	2.332	3.562	2.033	2.789	4.754

Table C.7: tRNA (guanine-N1-)-methyltransferase docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	4.816	-4.630	-5.483	4.569	4.380	5.221
std	2.317	2.116	3.164	2.160	1.988	2.920
min	0.739	-8.103	-12.541	0.927	1.037	0.302
25%	3.072	-5.340	-7.865	2.934	3.117	2.908
50%	4.783	-4.635	-5.340	4.528	4.377	5.042
75%	6.414	-3.284	-3.057	6.068	5.043	7.436
max	9.502	-1.050	0.274	8.960	7.649	11.835

Table C.8: tRNA (guanine-N1-)-methyltransferase redocked metrics/Å

Test set

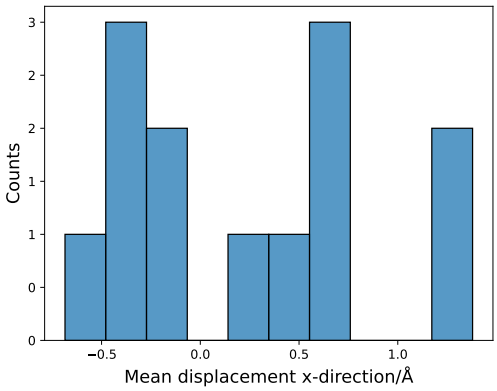
ABC transporter, periplasmic substrate-binding protein

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	0.278	0.461	0.073	0.834	1.032	1.142
std	0.667	0.830	1.479	0.427	0.862	1.211
min	-0.685	-0.587	-3.991	0.325	0.255	0.270
25%	-0.290	-0.169	-0.212	0.506	0.423	0.483
50%	0.174	0.440	0.197	0.795	0.691	0.623
75%	0.731	0.919	0.510	0.911	1.311	1.096
max	1.379	2.020	2.717	1.819	3.221	4.252

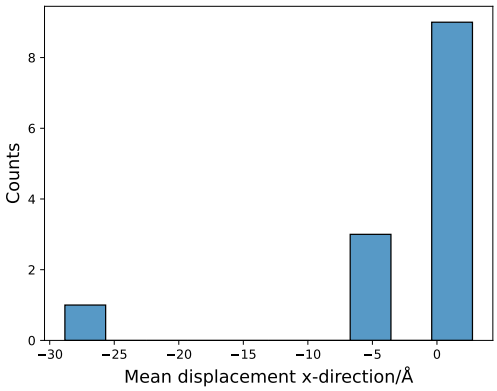
Table C.9: ABC transporter, periplasmic substrate-binding protein docked metrics/Å

	x-mean	y-mean	z-mean	x-std	y-std	z-std
mean	-2.526	-9.396	-20.738	3.859	12.744	19.688
std	8.334	30.450	33.828	7.199	27.081	31.804
min	-28.824	-107.842	-103.049	0.168	1.882	0.297
25%	-3.676	-12.826	-51.909	0.758	2.871	0.556
50%	0.619	2.648	-0.924	1.381	3.026	0.884
75%	1.358	3.188	-0.426	3.466	12.097	48.941
max	2.752	4.118	0.208	27.177	101.676	97.156

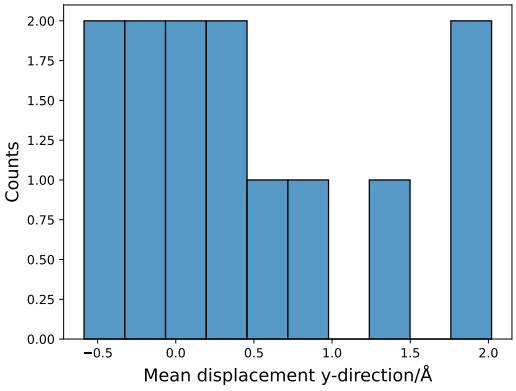
Table C.10: ABC transporter, periplasmic substrate-binding protein redocked metrics/Å



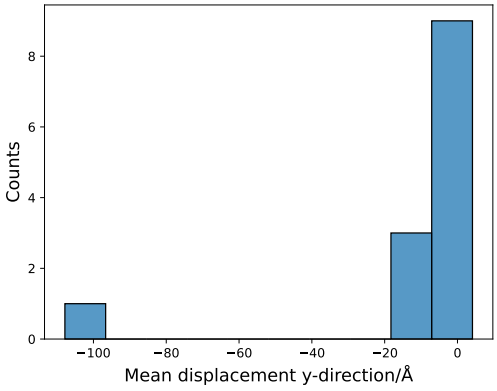
(a) Docked



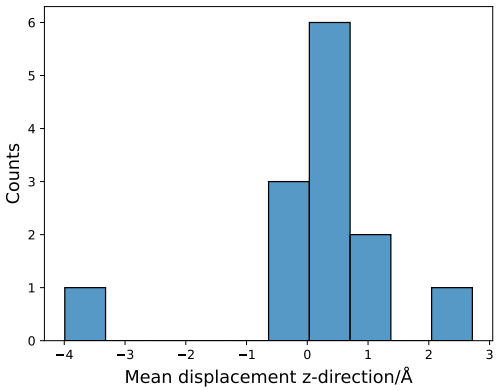
(b) Redocked



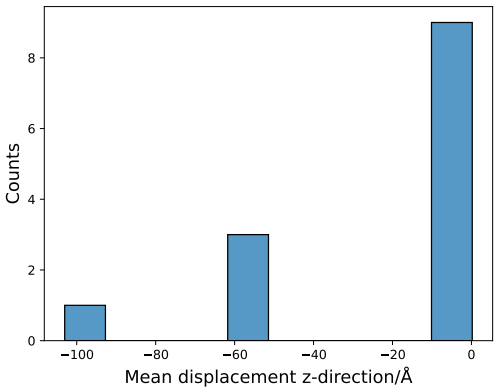
(c) Docked



(d) Redocked

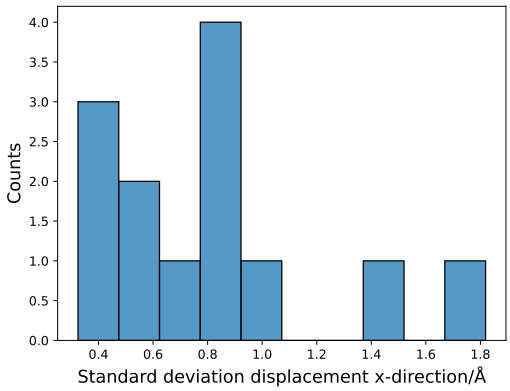


(e) Docked

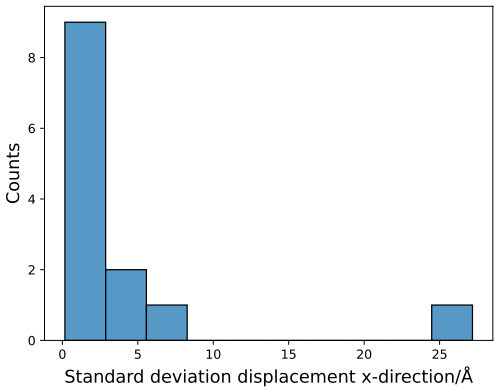


(f) Redocked

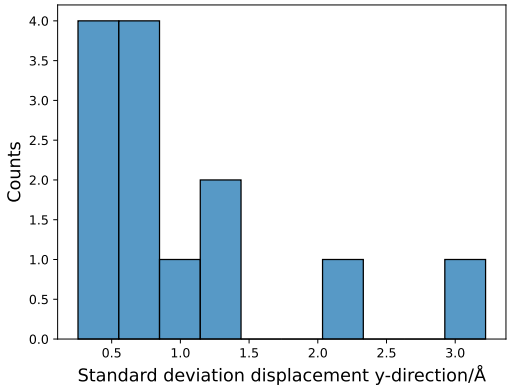
Figure C.9: ABC transporter, periplasmic substrate-binding protein mean displacements



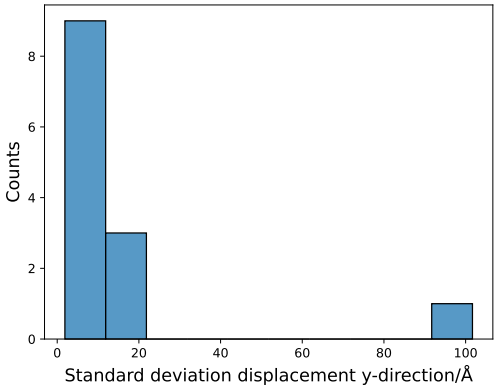
(a) Docked



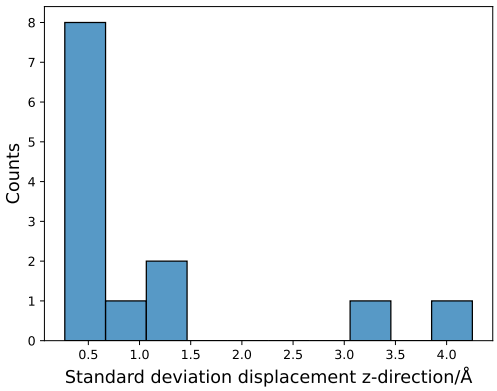
(b) Redocked



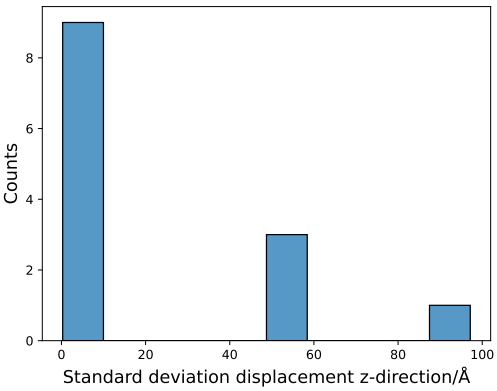
(c) Docked



(d) Redocked



(e) Docked



(f) Redocked

Figure C.10: ABC transporter, periplasmic substrate-binding protein standard deviation displacements

Appendix D

Antibiotics Hits

The SMILES strings for the 75 ligands used in this experiment are shown below:

1: O=C(Nc1ccccc1)C1CCC1#O=C(Nc1ccccc1)C1CCC1
2: N#N#Cc1cccc(Nc2ccccn2)c1
3: COc1cccc(Nc2ncccc2C)c1#COc1cccc(Nc2ncccc2C)c1
4: Cc1ccoc1Nc1cccc(C#Cc1ccoc1Nc1cccc(C#N)c1
5: N#N#Cc1cccc(Nc2nccs2)c1
6: CCC(=O)Nc1cc(C#CCC(=O)Nc1cc(C#N)ccc1C
7: CCc1ccc(O)c(CC(=O)NC)c1#CCc1ccc(O)c(CC(=O)NC)c1
8: CCc1ccc(O)c(CC2CCO2)c1#CCc1ccc(O)c(CC2CCO2)c1
9: N#N#Cc1ccc(O)c(Nc2ccco2)c1
10: Cc1nc2c(C#Cc1nc2c(C#N)ccc(O)c2[nH]1
11: CCc1cccc2[nH]c(C)nc12#CCc1cccc2[nH]c(C)nc12
12: N#N#Cc1cccc2[nH]c(=O)oc12
13: CC(C)C(=O)Nc1ccc(C#CC(C)C(=O)Nc1ccc(C#N)s1
14: Cc1ccc(NC(=O)C2CCCC2)s1#Cc1ccc(NC(=O)C2CCCC2)s1
15: CCCCC(=O)Nc1ccc(C)s1#CCCCC(=O)Nc1ccc(C)s1
16: C[C@H](C[NH3+])C(=O)Nc1cc(C#C[C@H](C[NH3+])C(=O)Nc1cc(C#N)ccc1O
17: CC(C)(C[NH3+])C(=O)Nc1cc(C#CC(C)(C[NH3+])C(=O)Nc1cc(C#N)ccc1O
18: COc1ccc(O)c(NC(=O)C(C)(C)C[NH3+])c1#COc1ccc(O)c(NC(=O)C(C)(C)C[NH3+])c1
19: COc1cnnc(NC(=O)CC[NH3+])c1#COc1cnnc(NC(=O)CC[NH3+])c1
20: COc1cnnc(CC2CC([NH3+])CCO2)c1#COc1cnnc(CC2CC([NH3+])CCO2)c1
21: N#N#Cc1cnnc(CC2CC([NH3+])CCO2)c1
22: N#N#Cc1ccnc(CC2CC([NH3+])CCO2)c1
23: N#N#Cc1ccnc2oc(CC[NH3+])cc12
24: CCc1ccc(O)c2oc(CC[NH3+])cc12#CCc1ccc(O)c2oc(CC[NH3+])cc12
25: C#C#Cc1cccc2oc(CC[NH3+])nc12
26: CCN(CC)c1ccc(C#CCN(CC)c1ccc(C#N)cc1
27: CCN(C)c1ccc(C#CCN(C)c1ccc(C#N)cc1O
28: CCC(=O)Nc1cc(C#CCC(=O)Nc1cc(C#N)ccc1N(C)CC
29: CCc1ccc(C[NH3+])[nH]c1=O#CCc1ccc(C[NH3+])[nH]c1=O
30: CCc1cc(NC(=O)C2CC2)c(C[NH3+])[nH]c1=O#CCc1cc(NC(=O)C2CC2)c(C[NH3+])[nH]c1=O
31: CCc1ccc(CO)[nH]c1=O#CCc1ccc(CO)[nH]c1=O
32: CCc1cc(NC(=O)C2CC2)c(CO)[nH]c1=O#CCc1cc(NC(=O)C2CC2)c(CO)[nH]c1=O
33: C#C#Cc1c[nH]c(CO)c1NC(=O)C1CC1
34: CCc1c[nH]c(N2CCNCC2)c1#CCc1c[nH]c(N2CCNCC2)c1
35: CCc1ccc(C2CC[NH2+]CC2)cc1#CCc1ccc(C2CC[NH2+]CC2)cc1
36: CCc1ccc(C2CC[NH2+]CC2)nn1#CCc1ccc(C2CC[NH2+]CC2)nn1
37: CCc1ccc(C2CCOCC2)nn1#CCc1ccc(C2CCOCC2)nn1
38: CCc1cc(CC(=O)N(C)C)c(C2CCOCC2)nn1#CCc1cc(CC(=O)N(C)C)c(C2CCOCC2)nn1
39: CCc1cc(CC(=O)N(C)C)c(C2CCOCC2)s1#CCc1cc(CC(=O)N(C)C)c(C2CCOCC2)s1
40: CCc1ccc(C2CCOC2)c(OC(=O)C(C)C)c1#CCc1ccc(C2CCOC2)c(OC(=O)C(C)C)c1

41: CC(C)C(=O)Oc1cc(C#CC(C)C(=O)Oc1cc(C#N)ccc1C1COC1
42: Cc1cnc(Cc2cc(C#Cc1cnc(Cc2cc(C#N)ccc2C2COC2)o1
43: Cc1cnc(Cc2cc(C#Cc1cnc(Cc2cc(C#N)ccc2C2CC[NH2+]CC2)o1
44: N#N#Cc1ccc(C2CC[NH2+]CC2)c(Cc2c[nH]nn2)c1
45: CCc1ccc(N2CC[NH2+]CC2)c(Cc2c[nH]c(C)n2)c1#CCc1ccc(N2CC[NH2+]CC2)c(Cc2c[nH]c(C)n2)c1
46: COCCc1cc(C#COCCc1cc(C#N)ccc1N1CC[NH2+]CC1
47: COCCc1cc(C#COCCc1cc(C#N)ccc1C[NH3+]
48: C[NH2+]Cc1ccc(C#C[NH2+]Cc1ccc(C#N)cc1CCSC
49: C[NH2+]Cc1ccc(Cl)cc1CCSC#C[NH2+]Cc1ccc(Cl)cc1CCSC
50: CCOCCc1cc(C#CCOCCc1cc(C#N)ccc1C(=O)NC
51: O=C(Nc1ccc(Cl)cc1)C1CC[NH2+]CC1#O=C(Nc1ccc(Cl)cc1)C1CC[NH2+]CC1
52: N#N#Cc1cc(NC(=O)C2CC[NH2+]CC2)ccc1Cl
53: N#N#Cc1cccc2[nH]c(C3CC[NH2+]CC3)nc12
54: N#N#Cc1cc(NC(=O)C2CC[NH2+]CC2)nnc1Cl
55: N#N#Cc1cc(Nc2csc3c2CC[NH2+]C3)nnc1Cl
56: N#N#Cc1cc(Nc2cc3c(s2)C[NH2+]CC3)nnc1Cl
57: N#N#Cc1cc(NCCCC[NH2+])ncc1Cl
58: Cc1cnc(NCCCC[NH2+])cc1C#Cc1cnc(NCCCC[NH2+])cc1C#N
59: Cc1cnc(NC(=O)CCC[NH2+])cc1C#Cc1cnc(NC(=O)CCC[NH2+])cc1C#N
60: Cc1cnc(CC(=O)NCC[NH2+])cc1C#Cc1cnc(CC(=O)NCC[NH2+])cc1C#N
61: Cc1cnc(NC(=O)NCC[NH2+])cc1C#Cc1cnc(NC(=O)NCC[NH2+])cc1C
62: Cc1cc(O)c(NC(=O)NCC[NH2+])cc1C#Cc1cc(O)c(NC(=O)NCC[NH2+])cc1C
63: N#N#Cc1cc(NC(=O)NCC[NH2+])c(O)cc1F
64: N#N#Cc1cc(CC(=O)NCC[NH2+])[nH]c(=O)c1F
65: N#N#Cc1cc(CC(=O)N2CC([NH2+])C2)[nH]c(=O)c1F
66: [NH2+]C1CN(C(=O)Cc2ccccc2O)C1#[NH2+]C1CN(C(=O)Cc2ccccc2O)C1
67: [NH2+]C1CC(C(=O)Nc2ccc(Cl)cc2O)C1#[NH2+]C1CC(C(=O)Nc2ccc(Cl)cc2O)C1
68: [NH2+]C1CCC(=O)N(c2ccc(Cl)cc2O)C1#[NH2+]C1CCC(=O)N(c2ccc(Cl)cc2O)C1
69: [NH2+]C1CCC(=O)N(c2ccc(Cl)cn2)C1#[NH2+]C1CCC(=O)N(c2ccc(Cl)cn2)C1
70: N#N#Cc1cc(N2CC([NH2+])CCC2=O)ncc1Cl
71: N#N#Cc1cc(N2C(=O)CC(C[NH2+])CC2=O)ncc1Cl
72: N#N#Cc1ccc(O)c(N2C(=O)CC(C[NH2+])CC2=O)c1
73: N#N#Cc1cc(N2CC(=O)N(CC[NH2+])CC2=O)ccc1Cl
74: N#N#Cc1cc(N2C(=O)CCC2CC[NH2+])ccc1Cl
75: [NH2+]CCCC(=O)Nc1ccc(Cl)cc1#[NH2+]CCCC(=O)Nc1ccc(Cl)cc1

Appendix E

Decoys

We now show the actives and receptors chosen for the decoys experiment in Section 4.5. For completeness we also list those actives which were removed due to failures either in processing the decoys SMILES into SDF files with Open Babel or running the SDF files through EquiBind-score.

Serine/threonine-protein kinase AKT

The target has PDB code 3cqw.

Actives

1. c1cc2cnccc2cc1c3cc(cnc3)OC[C@H](CC4CCCCC4)N
2. c1ccc2c(c1)cc(cn2)C[C@@H](COc3cc(cnc3)c4ccc5cnccc5c4)N
3. c1ccc2c(c1)c(cs2)C[C@@H](COc3cc(cnc3)c4ccc5cnccc5c4)N
4. c1ccc2c(c1)c(n[nH]2)C[C@@H](COc3cc(cnc3)c4ccc5cnccc5c4)N
5. c1ccc2cc(ccc2c1)C[C@@H](COc3cc(cnc3)c4ccc5cnccc5c4)N
6. c1ccc(cc1)C[C@@H](COc2cc(cnc2)c3ccc4cnccc4c3)N
7. c1ccc(cc1)[C@@H](COc2cc(cnc2)c3ccc4cnccc4c3)N
8. c1cc(c(c(c1OP(=O)(O)O)OP(=O)(O)O)OP(=O)(O)O)OP(=O)(O)O
9. c1c[nH]c2c1c(ccn2)c3c(cc(s3)C(=O)N[C@@H](Cc4ccncc4)CN)Br
10. Cc1c2cc(ccc2[nH]n1)c3cc(cnc3)NC[C@H](Cc4ccccc4)N

Beta-lactamase

The target has PDB code 1l2s.

Actives

1. S(Nc1c(O)cc(C(=O)O)cc1)(c2c(scc2)C(=O)O)(=O)=O
2. s1cccc1NS(c2c(C(=O)O)c(oc2C)C)(=O)=O
3. s1ccc(S(=O)(Nc2c(N3CCCCC3)cccc2)=O)c1C(=O)O
4. OC(=O)[C@@H](CC1=CC=CC2=CC=CC=C12)N1C(=O)C2=C(C=C(C=C2)C(O)=O)C1=O
5. S(=O)(c1c(C(=O)O)scc1)(NCc2ccccc2)=O

Fails

1. CC(C)CNC(=O)[C@@H]1[C@H]2C[C@@H]([C@@H]1C(=O)[O-])C=C2

2. C(=O)(c1c(cccc1)C(=O)O)Nc2ccc(Cl)cc2
3. c1ccc(cc1)[C@H]2C(=O)N(C(=O)O2)CC(=O)[O-]
4. s1c(C(=O)O)c(cc1)NS(=O)(=O)c2cccc(N(=O)=O)c2
5. OC(=O)C[C@@H](CC1=C2C=CC=CC2=CC=C1)N1C(=O)C2=C(C=C(C=C2)C(O)=O)C1=O

Glucocorticoid receptor

The target has PDB code 3bqd.

Actives

1. C[C@]12C[C@@H]([C@H]3[C@H]([C@@H]1CC[C@@]2(C(=O)CO)O)CCC4=CC(=O)C=C[C@]34C)O
2. CC#C[C@@]1(CC[C@@H]2[C@@]1(C[C@@H](C3=C4CCC(=O)C=C4CC[C@@H]23)c5ccc(cc5)N(C)Cc6cccc(c6)C(=O)C
3. CC1=CC(Nc2c1c3c(cc2)-c4c(cccc4SC)OC3CC=C)(C)C
4. CC1=CC(Nc2c1c3c(cc2)-c4c(cccc4OC3CC=C)C#N)(C)C
5. CC(C)(C)c1ccc(cc1)S(=O)(=O)N2CCC3=CC(=O)CCC3(C2)Cc4ccc(cc4)C#N
6. c1ccc(cc1)C[C@@]23CCC(=O)C=C2CCc4c3ccc(c4)O

Fails

1. c1ccc(cc1)S(=O)(=O)N2CCC3=Cc4c(cnn4c5ccc(cc5)F)C[C@@]3(C2)CN6CCOCC6
2. C[C@@]1(CC2(c3ccccc3C1c4c2cccc4)C#N)C(=O)Nc5nccs5
3. C[C@](c1ccc(cc1)F)(c2c[nH]c3c2cccc3NS(=O)(=O)C)C4CC4
4. CC(=O)Nc1ccc(cc1)S(=O)(=O)N2CCC3=CC(=O)CCC3(C2)Cc4ccccc4

Human immunodeficiency virus type 1 protease

The target has PDB code 1xl2.

Actives

1. c1ccc(cc1)C[C@@H]2[C@@H]([C@H]([C@H](N(C(=O)N2Cc3ccc(cc3)CO)Cc4ccc(cc4)CO)Cc5ccccc5)O)O
2. c1ccc(cc1)C[C@@H]2[C@@H]([C@H]([C@H](N(C(=O)N2CC3CCC3)CC4CCC4)Cc5ccccc5)O)O
3. CC(C)CCN(C[C@H]([C@H](Cc1ccccc1)NC(=O)[C@H](CC(=O)N)NC(=O)c2ccc3ccccc3n2)O)S(=O)(=O)C
4. Cc1ccc(cc1)S(=O)(=O)N(CC(C)C)[C@@H](CCCCNC(=O)COc2ccccc2)C(=O)O
5. CCCCCC(=O)N[C@@H](CC1CCCCC1)[C@H](C[C@@H](C(C)C)C(=O)N[C@@H](C(C)CC)C(=O)NCc2ccccn2)O
6. c1ccc(cc1)C[C@@H]2[C@@H]([C@H]([C@H](N(C(=O)N2Cc3ccccn3)Cc4ccccn4)Cc5ccccc5)O)O
7. c1ccc(cc1)CCC[C@@H]2[C@@H]([C@H]([C@H](N(C(=O)N2Cc3ccccc3)Cc4ccccc4)CCCc5ccccc5)O)O

Fails

1. CCCCN1[C@@H]([C@@H]([C@H]([C@H](N(C1=O)Cc2cccc(c2)C(=O)Nc3[nH]ccn3)Cc4ccccc4)O)O)Cc5ccccc5
2. c1cc(cc(c1)NS(=O)(=O)c2cscn2)C(c3c(=O)c4c(oc3O)CCCCCC4)C5CC5
3. c1cc(cc(c1)NS(=O)(=O)c2cccc(c2)C(=O)O)C(c3c(c4c(oc3=O)CCCCCC4)O)C5CC5