COMPARISON OF TRADITIONAL AND MACHINE LEARNING PROGRAMS IN THE EVALUATION OF PROTEIN-LIGAND BINDING

Blessing Anyangwe, Arushi Desai, Elizabeth Fishman, Kevin Jin, Kai Kim, Erin Kraus, Eugene Lee, Angelina Li, Bridget Liu, Nicholas Sardy, Aarna Tekriwal, Osariemen Unuigbe, Alexander Zatuchney, Eric Zhu

Advisor: David Cincotta
Assistants: Katerina Pouathas, Joel Moses

ABSTRACT

Molecular docking is an *in-silico* method that predicts the conformation of an interaction between two or more molecules—generally, an interaction between a ligand and its target protein. In recent years, molecular docking has had widespread applications in pharmaceutical fields, aiding in the discovery and development of drugs. Traditionally, molecular docking is performed using a search and score algorithm: the search algorithm generates a variety of protein-ligand poses, and the scoring algorithm calculates the binding strength of each pose. These results can then be used to determine the optimal binding conformation of a protein-ligand complex. More recently, molecular docking programs that rely on artificial intelligence (AI) have been developed. These deep-learning-based models—based on reverse diffusion—are trained with datasets of protein-ligand complexes. By analyzing existing protein-ligand complexes, these deep-learning molecular docking programs progressively become more successful at docking ligands to their target proteins given their molecular properties. To compare the accuracy of traditional and deep-learning-based molecular docking programs in site-specific docking (i.e. "pocket docking"), a traditional program and a deep-learning-based program were tasked with docking a set of ligands to their target proteins—all of which were not included in the deep-learning-based program's training datasets. The traditional docking program of choice was SeeSAR "Midas" (v.13.1.1) by BioSolveIT; the deep-learning-based docking program of choice was DiffDock-Pocket, a pocket-docking version of the standard blind-docking DiffDock program. After using both programs to predict the protein-ligand binding conformations, the Root Mean Square Deviation (RMSD) value of each predicted protein-ligand conformation relative to the actual protein-ligand conformation was determined using a Python program. The RMSD values generated by the two docking programs (see Appendix B)—used to quantify docking accuracy—were then compared for all tested protein-ligand complexes. After comparing the docking accuracies of the two programs, it was determined that SeeSAR "Midas" and DiffDock-Pocket predict optimal protein-ligand binding conformations with relatively similar accuracies. Especially given the novelty of AI-based approaches in molecular docking, these methods are especially promising, and it is possible that they may surpass traditional docking programs in the near future.

INTRODUCTION

The objective of this project was to assess and compare the accuracy of traditional and deep-learning molecular docking technologies to determine the value of machine learning programs within the realm of molecular docking and drug discovery.

Protein-Ligand Binding

Traditionally, protein-ligand complexes were thought to be static structures with perfect geometric complementarity—analogous to a key within a lock—as proposed by chemist Emil Fischer in 1894 (Nelson et al. 2021). This "lock and key" hypothesis, however, was later determined to be energetically implausible, especially when applied to enzyme catalysis—which largely overlaps with protein-ligand interaction (Bankaitis and Tripathi 2017). Since enzymes reduce the activation energy necessary for a substrate to reach its transition state, an enzyme that is most geometrically compatible with its substrate in its initial state would stabilize the substrate's initial structural conformation, preventing it from reaching the transition state and thus undermining enzyme catalysis (Nelson et al. 2021). Instead of being most complementary to a substrate's initial state, enzymes are most complementary to a substrate's transition state, and weak binding interactions must continuously form between the enzyme and substrate upon initial binding to lower the activation energy barrier. Thus, enzyme-substrate interactions are dynamic and continue to tighten after initial binding—a model called induced fit that is commonly compared to a "handshake" (Urry et al. 2020). The principles of the induced fit model extend to all protein-ligand interactions, which are also dynamic structures that must perform adjustments to reach an optimal binding conformation (Nelson et al. 2021). Molecular docking is a computational technique that has been developed that uses the idea of induced fit interactions to predict optimal binding conformations between proteins and ligands, often for pharmaceutical purposes.

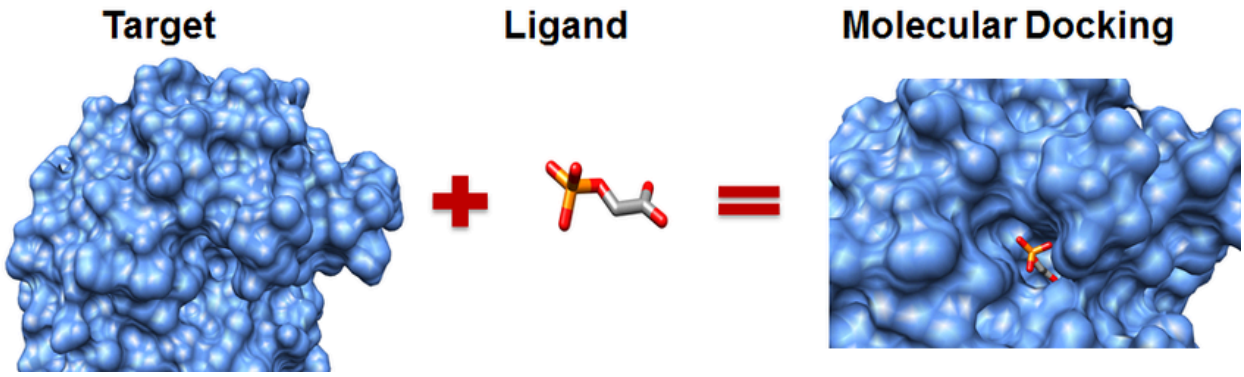

Fig. 1: Basic molecular model describing protein-ligand interactions in biological systems (Scigenics from *ResearchGate* 2015).

Definition and Significance of Molecular Docking

Molecular docking is an *in-silico* method in bioinformatics that computationally assesses the interaction of two or more molecules and is assessed based on the stability and strength of the interaction (Agarwal et al. 2021). Docking programs are used to predict the optimal binding conformation of a small protein-ligand complex given a ligand and its respective target protein.

Thus, the primary goal of molecular docking is to accurately identify the most energetically favorable orientation of a protein-ligand complex and present the appropriate binding affinities of a specific ligand within a binding site of a target protein. Though many validation and scoring methods exist, a commonly used method is pose selection. In pose selection, a docking program is tasked with re-docking a ligand with a known pose. A successful prediction falls within a chosen threshold Root Mean Square Deviation (RMSD) value (usually below 2Å). After pose selection, scoring and ranking occur, generating a list of the most accurate pose predictions (Hevener et al. 2009).

Molecular docking has significantly evolved from its first appearance in the mid-1970s. Since then, it has become an essential part of determining the binding conformations of macromolecules to their molecular targets (Pinzi and Rastelli 2019). Now commonly used in drug development, it has become a crucial tool in the medical field for curing diseases and treating symptoms because of its ability to predict possible ligand-DNA interactions. More specifically, molecular docking has become an integral part of drug discovery and optimization because a key component of structure-based drug optimization is ensuring that a drug of interest can bind to specific binding sites of a targeted protein in a conformation that allows it to fulfill its physiological purpose.

Particularly, molecular docking has become important in predicting the activity and efficacy of new pharmaceutical drugs by predicting the structural complementarity and energetic favorability of a complex consisting of a drug and its target molecule. This can potentially identify unexpected binding conformations between drugs and target molecules, thereby allowing researchers to minimize unwanted side effects—especially life-threatening ones. Additionally, molecular docking has crucial applications in the field of polypharmacology, where it is used to identify ligands that simultaneously bind to a pool of selected targets of interest. Finally, docking has been used to determine novel uses for chemical compounds with already optimized safety profiles, an approach called drug repositioning (Pinzi and Rastelli 2019). The widespread pharmacological applications of molecular docking have directed significant attention towards molecular docking.

Molecular Docking: Traditional Methods

Traditionally, molecular docking techniques rely on a search-score algorithm, which generates various protein-ligand binding conformations, or poses, and assesses the binding strengths of these poses. Within the last decade, the demand for competent docking programs has increased due to the speed and convenience of these programs (Torres et al. 2019). The docking process begins with obtaining a target structure, typically a biological macromolecule such as DNA or a protein. A majority of docking programs, including SeeSAR, have built-in access to the Protein Data Bank (PDB), which provides information about proteins in addition to their three-dimensional structures–which have already been characterized with a laboratory technique (i.e. X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, cryo-electron microscopy (cryo-EM)) (Nelson et al. 2021). A ligand or separate molecule is then selected for docking, taking into account the torsion of the ligand. However, a limitation of the traditional search-based molecular docking technique is that it is unable to account for the significantly increased dimensionality of the search space that occurs with protein flexibility (Yu et al. 2023;

Pinzi and Rastelli 2019). This limitation inhibits the efficient production of drug design in current markets.

Traditional molecular docking programs use a variety of search algorithms to determine the optimal position for a ligand binding with a protein. In a conformational search process, ligands are progressively rotated and observed in their binding conformation. In fragmentation, the ligand is separated into fragments, which are individually bound to the macromolecule, and then repositioned and combined to find the most favorable position. Programs may also use smaller bound structures stored in their database to estimate the binding position of similar, larger molecules. Other processes, such as approaches based on the Monte Carlo algorithm, randomly insert ligands to determine the optimal position. A genetic algorithm-based approach utilizes a pose and then adds mutations, or random changes, which it then analyzes and ranks in terms of compatibility. Tabu searches analyze the failures of previously identified binding attempts to apply further restrictions in new binding simulations. Programs like “LUDI” and “DOCK” use a fragmentation-based method while “MCDOCK” uses a Monte Carlo approach and “AutoDock” and “GOLD” use a genetic algorithm (Xu et al. 2018).

Molecular Docking: Deep Learning Methods

More recently, machine learning-based molecular docking has been developed due to potential optimizations in time and ease of access. These deep-learning-based (DL) methods—which utilize reverse diffusion models—do not rely on conventional search-score algorithms and have improved processing speeds and overall optimize the molecular docking process. Machine learning algorithms use training data to “learn,” predict binding patterns, and properly generalize unseen information; they are essentially controlled by their training data because they are “knowledge-based” rather than physics-based and are entirely based on included input data (Balius and Rigby 2022).

This learning is specifically done by the use of scoring functions to evaluate and rank the variety of ligand binding poses that are possible for a specific protein-ligand complex. Scoring functions rank these various ligand-binding poses based on a variety of factors. These factors include electrostatic, polar, nonpolar, hydrophilic, and hydrophobic interactions within the protein-ligand complex as well as how the complex interacts with its environment or “residues" it establishes interactions with (Qing et al. 2022). Scoring functions represent a particular ligand binding pose’s validity within the protein-ligand complex. Thus, machine learning algorithms are typically provided with multiple datasets, which typically include the known binding affinities, structural information, and interactions of provided protein-ligand complexes (Libouban 2023). The purpose of analyzing these protein-ligand complexes is to learn the different interactions that occur between proteins, ligands, and their surrounding residues. These molecular interactions are then stored in the hard drive of the machine learning algorithm.

Given these initial training datasets, the machine learning algorithms learn from their training datasets to predict binding properties (among them being binding affinity) of specific protein-ligand complexes, based on their molecular composition (Qing et al. 2022). However, it must be said that accuracy of specific protein-ligand complex binding properties can vary, depending on the datasets the machine learning algorithms use versus the actual protein-ligand

complexes they are given to evaluate (Qing et al. 2022). To put this into perspective, if a protein-ligand complex is inputted into a deep-learning algorithm, in which virtually none of the training protein-ligand complexes share any compositional or structural similarity, the deep-learning model would have nothing to base its molecular composition on to determine any sort of binding property. It is reasonable to expect that the results of the given experimental protein-ligand complex would not be accurate to the actual known molecular interactions and specific properties (like specific binding sites and binding affinity) of the experimental protein-ligand complex because its training set does not provide any referable binding spots or interactions or molecular composition data.

Comparing Traditional Methods and Deep Learning Methods

Traditional methods and deep-learning methods for molecular docking possess characteristic advantages and disadvantages. For instance, traditional methods tend to specialize in molecular docking processes in which the target protein's binding pocket has already been provided alongside the ligand and the target protein (called "site-specific docking") (Yu et al. 2023). In essence, traditional methods require a given pocket to dock on the entire protein (Yu et al. 2023). For this reason, traditional methods tend to outperform deep-learning programs in site-specific docking. However, many traditional docking methods experience complications in terms of accuracy and sampling space, especially when running particularly large protein-ligand complexes (Qing et al. 2022). Furthermore, traditional methods typically fail to consider protein flexibility and conformational changes. This may undermine the accuracy of its scoring functions, thereby preventing the most accurate binding affinities (and/or other properties) from being determined.

Deep learning methods, on the other hand, generally perform "blind-docking," a process that docks a ligand to an entire protein that involves both pocket searching and site-specific docking. Deep learning methods tend to be very competent in locating a given protein's pocket (called "pocket searching")—a necessary step in the blind-docking process (Yu et al. 2023). Moreover, the generative diffusion models used in deep-learning approaches offer the benefit of being able to better emulate the stochasticity of biological processes (Plainer et al. 2023). These generative diffusion models better capture the overall flexibility of molecular interactions in biological systems. As a result, many deep-learning models are able to account for specific complex interactions between a protein and its ligand that traditional docking methods may have failed to capture. It is important to note, however, that the performance of a deep-learning molecular docking program is dependent on the provided training datasets. Using similar proteins and ligands to those in the training dataset can potentially skew the docking performance of a DL-based docking program. Thus, when testing the docking capability of these programs, it is important to consider proteins and ligands distinct enough from those used to train the docking program. It is also important to recognize that the accuracy and effectiveness of deep-learning models are not standardized; deep-learning models require large training datasets, which are often difficult to obtain for drug optimization. Thus, depending on the training data (i.e., its similarity testing data) and the accuracy of a deep-learning model, calculations characterizing different protein-ligand interactions may vary. In contrast, docking remains relatively consistent when using traditional docking programs, especially for smaller protein-ligand complexes.

Currently, deep-learning models typically perform well with pocket searching while traditional docking methods produce better results when performing site-specific docking (Pinzi and Rastelli 2019). With the advent of new deep-learning software, discussions have been initiated comparing the accuracy of these programs relative to previously existing traditional programs. To compare the efficacy of traditional and deep-learning molecular docking technologies, selected traditional and deep-learning programs can be used to perform molecular docking on ligand-protein complexes with known optimal binding conformations.

BioSolveIT's SeeSAR 'Midas' (v. 13.1.1) is a "traditional" molecular docking software that performs site-specific docking. It uses search-score algorithms to evaluate various molecular interactions, including possible protein-ligand poses. SeeSAR comparatively assesses the energetic favorability of each pose. The energy of each pose is calculated by scoring functions to identify the pose with the least calculated energy (i.e., the most optimal binding conformation) (Plainer et al. 2023).

DiffDock-Pocket—a site-specific docking program developed to be an extension of the standard DiffDock program (Plainer et al. 2023)—is a molecular docking program that uses a diffusion-based, all-atom docking algorithm. DiffDock-Pocket uses "the often available prior knowledge about the binding site, which is where the ligand has the most significant effect on the protein structure" and it restricts "protein flexibility to the side chains interacting with the ligand while keeping the more rigid backbone atoms fixed" (Plainer et al. 2023). The program is given sets of known protein-ligand combinations and forms "the problem as a generative modeling task" and develops "a diffusion model jointly over the protein side chain torsion angles, the ligand torsion angles, and the relative position of the protein and the ligand" (Plainer et al. 2023). DiffDock-Pocket uses random ligand poses and side chain conformations along with a reverse diffusion process to predict realistic binding conformations (Plainer et al. 2023). Because DiffDock-Pocket has a site-specific approach, it can be compared to traditional methods more accurately. This deep-learning-based program requires the input of a ligand of interest placed within its target protein's pocket, which initially features randomly initialized side chains. As the docking process proceeds, DiffDock-Pocket repeatedly updates protein joint structures such that protein-ligand binding conformations become progressively more realistic. The program utilizes a confidence model to score and rank the generated protein-ligand conformations, allowing the most plausible conformations to be determined (Plainer et al. 2023).

Given this, it is hypothesized that the traditional method (SeeSAR) will be more accurate than the machine learning program (DiffDock-Pocket) because of the consistent, physical basis of the traditional docking programs.

METHODS

Dataset Formation

An ideal dataset to test molecular docking programs contains proteins that do not share similar characteristics to proteins in their dataset. Many existing datasets, however, do not meet this requirement, as non-homologous structures may still have similar binding pockets due to

pockets' highly conserved nature. Corso et al. (2024) sought to solve this issue when evaluating their DiffDock-L program. Many machine learning protein dockers, like DiffDock and its derivatives, utilize the PDBBind dataset for their training data (Liu et al. 2017). To test their model, Corso et al. utilized the Binding Dataset from the Mother of All Databases (MOAD), which utilizes different criteria to form its dataset compared to PDBBind, to make a 189 protein dataset called Docking Generalization, more commonly referred to as "DockGen" (Corso et al. 2024). Corso et al. found that existing machine learning programs had a large drop in accuracy with this dataset, showing that it is sufficiently different enough from their training datasets to use in generalization testing (Corso et al. 2024).

DockGen was used as a base and then reduced by removing protein-ligand pairs without structure quality assessments present in the RCSB Protein Data Bank (PDB) for a resulting dataset of 135 protein-ligand pairs. Ground truth ligand pairs can be defined as ligand pairs that are formed from experimental crystallographic data. The dataset was then accumulated into separate files and codes corresponding to differing programs and inputted into SeeSAR and DiffDock-Pocket for analysis.

Traditional Docking

Molecular docking was conducted with the proteins and ligands in the dataset using both SeeSAR v. 13.1.1 and DiffDock-Pocket.

The SeeSAR traditional molecular docking program was used to predict the ligand position of the protein-ligand combinations found in the data set. First, a protein from the dataset was loaded into SeeSAR. As a note, SeeSAR always loads the asymmetric unit of a protein, which means that the same biological assembly was used for the ground truth and DiffDock-Pocket. When a protein loads into SeeSAR, a list of ligands is provided. The first ligand that has an identical name to the one being analyzed was extracted, maintaining its binding pocket, and then added to the molecule editing mode. Afterward, the ligand was deleted to make the pocket unoccupied, and a SMILES code of the same ligand was added to the protein through the Analyzer mode. Finally, that new molecule, which has not had its pose within the now-empty pocket calculated, is added to the protein docking mode, and standard docking was performed to generate a pose for the ligand being analyzed. When the docking completes, a list of calculated poses and their estimated binding affinities is displayed. The pose with the lowest binding affinity (and thus the pose that is most likely to bind in the pocket) was saved as a .sdf file, and the protein-molecule complex was also saved for further analysis. These files were then compared to the ground truth files found on PDB to calculate their RMSD.

Machine Learning Docking

The Machine Learning program, DiffDock-Pocket, was also used to predict proper ligand position in a protein ligand interaction. These ligands were cross-referenced with the dataset and the protein-ligand structures used in SeeSAR data collection to create a comparison between SeeSAR and DiffDock-Pocket. The asymmetric structures of each protein were downloaded as PDB files from the Protein Data Bank to match the biological assembly used in SeeSAR. The proteins were then fixed to remove all bound ligands, the aqueous solvent, and all other residues

so that the protein could be analyzed without any interference. Then, the ligand being tested in the dataset was extracted from the PDB files using Python scripts (see Appendix A) to get the ground truth ligand provided by the Protein Data Bank as well as the proteins being analyzed, that are not occupied by the ligand. DiffDock-Pocket ran the unoccupied proteins and exported .sdf files of the ligand that was used. The rank 1 ligand file was used in comparison to the ground-truth ligand file, producing the RMSD value.

Calculation of Data

Evaluating the molecular docking through the traditional SeeSAR docking program and the Machine Learning DiffDock-Pocket docking program required a shared metric that could compare the ground truth ligand-protein complexes to the complexes generated by each program. This shared metric was Root Mean Square Deviation (RMSD), which "is the most commonly used quantitative measure of the similarity between two superimposed atomic coordinates" (Abagyan and Kufareva 2015). RMSD (see Equation 1) is calculated by measuring the distance between equivalent atoms in two superimposed structures. *n* represents the number of atom pairs and *di* is the distance between the two atoms. RMSD is 0 for identical structures and increases as the structures become more diverse. The RMSD values for each of the 135 protein-ligand pairs were calculated for both SeeSAR and DiffDock-Pocket using a Python-based program. To calculate these values, the data outputted by both programs was compared to the ground truth ligand position/orientation. The similarity of the data (the ligand poses) produced by each program and the ground truth ligand poses produced the RMSD values. Values below 2Å were considered to be reliable while values below 1.5 Å were considered very reliable.

$$RMSD = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2}$$

Equation 1: Equation of Root Mean Square Deviation (RMSD), the metric used to evaluate generated protein-ligand complexes of both SeeSAR and DiffDock with ground truth protein-ligand complexes (Kufareva and Abagyan 2015).

Protocol

The process of running the proteins on the two programs and calculating the RMSD values between the outputs and the ground truth ligands included various steps. First, the .pdb files were downloaded from the RCSB Protein Data Bank. A Python script was used to extract the corresponding ground-truth ligands from the .pdb files and convert them to .mol2 format. For

DiffDock-Pocket, another script was used to fix the .pdb files by removing hydrogens and adding missing atoms. Subsequently, both programs were run using the appropriate files. For DiffDock-Pocket, the fixed .pdb file and the ground-truth ligand in .mol2 format were inputted into the program. For SeeSAR, the protein data was collected through the program's built-in database and the proper ligand was inputted to the program in .pdb format.

The above procedure was followed for each protein in the dataset. Subsequently, the outputted .sdf files were gathered alongside the ground-truth ligands in .mol2 format, and Python code was written and used for calculating RMSD values for each program. (See Appendices A-1 and A-2 for the full scripts for DiffDock-Pocket and SeeSAR, respectively).

RESULTS

Appendix B contains all the RMSD values for each protein analyzed. 36 of the proteins analyzed did not output an RMSD value due to errors in both programs. To better compare the two programs' ability to predict the optimal binding position only the proteins that ran through the programs and the calculation were analyzed. To visually analyze and compare the RMSD values calculated for SeeSAR and DiffDock-Pocket, two histograms (Figure 2a and 2b) were created through the SPSS program. The X-axis contains the range of RMSD values grouped into 7 bins. Each bin has a range of 1 angstrom, starting from 0 angstroms to 6. Bin 7 contains the number of protein-ligand complexes that have an RMSD value above 6 angstroms. Table 1 represents the percentage of proteins that ran on SeeSAR and Diffdock-Pocket that were successful and unsuccessful at docking.

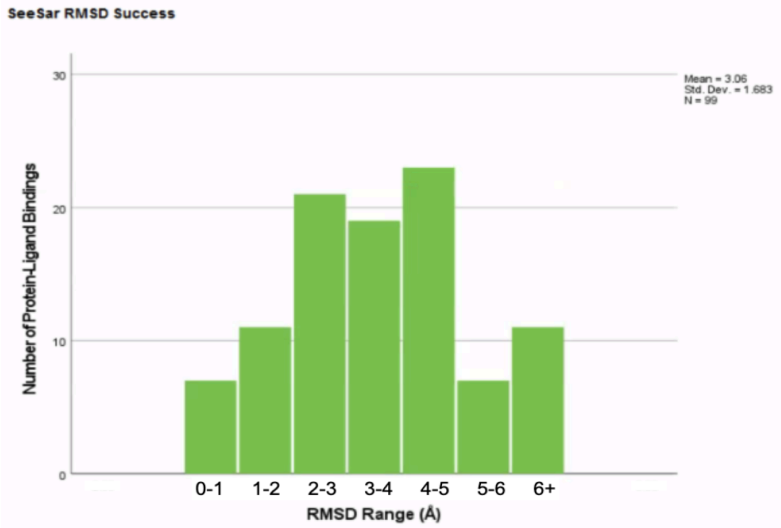

Fig. 2a: Histogram displaying the RMSD values of the proteins that successfully ran through SeeSAR and the calculations.

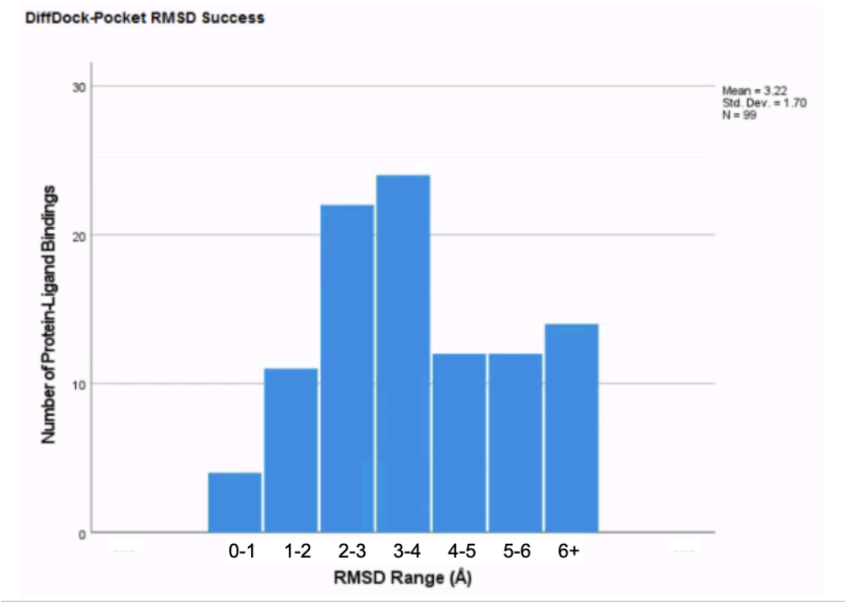

Fig. 2b: Histogram displaying the RMSD values of the proteins that successfully ran through DiffDock-Pocket and the calculations.

	SeeSAR	DiffDock-Pocket
%Successful (≤2Å)	18.2%	15.2%
%Unsuccessful (>2Å)	81.8%	84.8%

Table 1: Percentage of proteins that ran on both programs that were successful and unsuccessful at docking.

The results from this experimental data indicate that machine learning software such as DiffDock-Pocket output results on par with traditional computational programs such as SeeSAR. This is clear by looking at the outputs for RMSD values under 2Å indicating that both programs had a similar number of successful docking procedures. When looking specifically at the values that successfully ran, SeeSAR had a success rate of 18.2% (18 proteins), while DiffDock-Pocket had a success rate of 15.2% (15 proteins). Although DiffDock-Pocket had a 3% lower success rate, the difference between the success rates is not large enough to conclude that SeeSAR was significantly better in predicting successful dockings. Additionally, as per Figures 2a and 2b, the mean RMSD value and standard deviation have very little variation between DiffDock-Pocket and SeeSAR. Particularly, the difference in mean RMSD value is 0.16 and the difference in standard deviation is 0.017. This shows that each RMSD value calculated for the two programs is virtually the same for each run, meaning that there is not enough difference between the two programs to be able to determine which one is better.

However, one of the major discrepancies in the data between the two programs is the protein-ligand combinations that could not be run: SeeSAR had a larger number of protein-ligand

pairs for which an RMSD value could not be calculated compared to DiffDock-Pocket. Specifically, the number of proteins that were not able to be calculated from DiffDock-Pocket was 18 from the 135 protein-ligand pair dataset (13.3% of the total), while SeeSAR was unable to run 31 (23.0% of the total). Therefore, SeeSAR, although able to create a relatively high number of successful dockings, can generate a lower number of appropriate binding structures for analysis compared to DiffDock-Pocket.

DISCUSSION

SeeSAR was initially expected to produce smaller RMSD values given that it is a more reliable model; in this experiment, however, SeeSAR was less reliable than anticipated–especially in comparison to the newer, deep-learning-based DiffDock-Pocket because the produced RMSD values were much higher than the experimental ground truths. Moreover, obtaining RMSD values from SeeSAR required the user to perform more operational steps, unlike the automated DiffDock-Pocket program. This, in turn, also increased the possibility of human error.

DiffDock-Pocket performed remarkably better on the DockGen dataset when compared to other AI docking models. Corso et al. demonstrated in their study that other models like GNINA, SMINA, and Equibind had a success rate of 10.6%, 17.5%, and 0.0% respectively (Corso et al. 2024). Furthermore, the original DiffDock had a success rate of 6.0% and DiffDock-L had a success rate of 22.6%. DiffDock-Pocket, thus, places around average in terms of success rate. However, DiffDock-Pocket had the lowest median error out of all the programs, likely due to its pocket-specific approach, showing that the program has promise for docking prediction.

CONCLUSION

The purpose of this project was to evaluate the accuracy of machine learning-based Molecular docking programs in comparison to traditional programs. DiffDock-Pocket and SeeSAR had similar success rates and median error values. Though there is some slight variation in the magnitude of error, it is not significant enough to conclude that either program is more accurate. Based on the data collected and the analysis of the data, there is evidence that Molecular Docking programs that enlist the help of Machine Learning can be realistically considered in drug formation. Although the DiffDock-Pocket program did not have a significantly larger success rate than SeeSAR, AI and Machine Learning programs are growing exponentially and there is a strong possibility that they will be fit for quick and accurate protein-ligand combinations. This possibility aligns with the research conducted, as DiffDock-Pocket – the unreleased machine learning program – was on par with SeeSAR–the older, commercially released, and frequently updated program. As more training data is introduced into the machine learning programs, the accuracy and precision of the protein-ligand binding complexes will become better aligned with the already existing data on data banks such as the PDB. Additionally, as more users interact with the interface, the overall program becomes more accessible to all users, allowing for better troubleshooting in the future, and creating an overall better interface.

However, user discretion can be used to determine what software is better in the corresponding situations. If a user wants a more automated experience, they can refer themselves to using the DiffDock-Pocket software. Once the files are added in and are run, the RMSD values are automatically calculated and the user does not have to sort through the differing pockets. SeeSAR has a more manual interface, in which users must select the specific ligand in use and alter the molecule to run the protein. It is important to note that these programs both had similar running times and with increased usage of and familiarity with the program, the run time will decrease.

Limitations

RMSD is a common metric used in the scientific community and is the sole metric used in this study to compare DiffDock-Pocket with SeeSAR. Although RMSD is used repeatedly in scientific research and study, there are some limitations to how accurate comparisons can be by relying on one metric. To create the most valid comparison of the two docking programs and minimize error, optimally more than one metric would be compared, to account for other factors that evaluate molecular docking accuracy, such as binding affinities. With multiple metrics, the two programs could be compared on different levels of evaluation, providing the researchers with a more multidimensional analysis. With more data, the strengths and weaknesses of both programs could be found and the conclusion would be more accurate. However, access to experimental data that could be collected from both SeeSAR and DiffDock-Pocket was limited, so a more broad metric had to be used. The limitation of data limits the points of comparison available and prevents a more in-depth resulting claim.

While the RMSD values for the DiffDock-Pocket outputs were accurate, the script run for the SeeSAR data had a large error. Figure 3 represents the superimposition of the ligand in 3ZZS, which had a calculated RMSD of 2.76. Although the error may be attributed to the presence of hydrogen side chains in the SeeSAR calculated ligand, removing the hydrogens does not change the value. The error, thus, is likely due to a misalignment of atom order between the ground truth .mol2 file and the SeeSAR .sdf file. In reality, this ligand has an RMSD of .533, as found on zhanggroup.org. However, this website cannot be automated, and due to time constraints, the remaining ligands could not be compared using this resource. As such, it is important to note that the majority of RMSD values for the SeeSAR calculations are significantly larger than the actual RMSD values, and SeeSAR likely is more accurate at predicting binding pockets in generalization datasets.

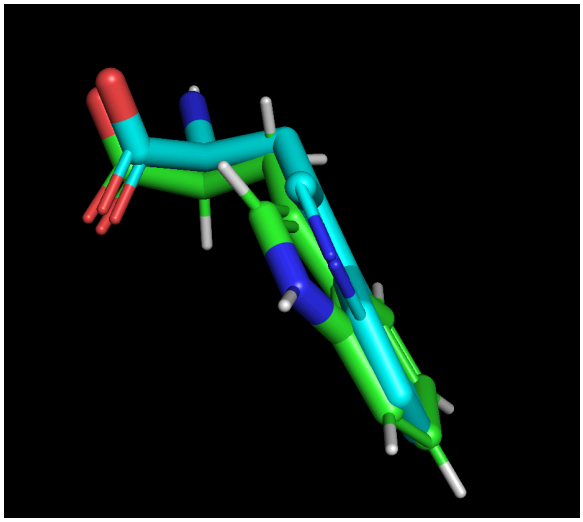

Fig. 3: the superimposition of the SeeSAR calculated 3ZZS TRP ligand (green) with the ground truth (blue)

Furthermore, RMSD values are biased. Due to the nature of the equation used to calculate RMSD, smaller molecules generally result in a lower RMSD value. Because this study compares a calculated protein-ligand combination to its ground truth, with the only variations being the ligand pose and marginal flexibility in the protein, the global protein RMSD values cannot be used accurately with the same threshold used for the ligand assessment. As such, the RMSD data collected is limited to only a comparison of the local ligand.

Finally, solely using RMSD has limited applications when considering binding affinity. Though the accuracy of the position of the ligand is important, if the ligand is not likely to bind in the pocket in the first place, the data may not be relevant. SeeSAR can calculate binding affinity for generated poses, but DiffDock-Pocket does not. Notably, other ML programs like GNINA can predict binding affinity from a confidence score, but GNINA struggles with the DockGen dataset, as demonstrated in Corso et al. 's paper (2024).

Of the 189 protein-ligand pairings initially available in the DockGen Dataset, only 135 were ultimately used due to a lack of available protein-ligand quality assessments for a portion of the protein-ligand pairs in the dataset.

In addition, analysis of protein-ligand interactions on SeeSAR was limited to only being completed on Windows, and DiffDock-Pocket only being completed on Mac. This was primarily due to DiffDock-Pocket currently being inaccessible on Windows. The use of these operating systems was not the most ideal in the context of the research. First and foremost, it would have been optimal to use a more powerful operating system product with high RAM storage capacity for the most optimal results on both DiffDock-Pocket and SeeSAR. Other than that, because the operating system used was not standardized, it may be possible for results from one of the molecular docking software to be generated that may not translate to the same value if the process of that same docking software were to be performed on the other operating system.

Only one pocket per protein was considered for evaluation, even though many proteins in the dataset had more than one available pocket for a molecule to bind. For example, 3ZZS has 8 pockets for TRP to bind, but only the first was considered. Both programs can evaluate more than one ligand, but due to time constraints, only the top pocket was evaluated.

The reliability of the programs can be evaluated by running the programs again on already-run proteins to determine if the RMSD value is the same. The results of this test found that the reliability of these programs is questionable; when the same protein-ligand interactions were run, the RMSD values were not identical to the ones that were already run. The SeeSAR program itself has reliability itself, however as it was run through Python code, the issue can be identified there. The Python code could have produced errors, changing the RMSD value that was originally found. The DiffDock-Pocket program, however, could have errors in it. As this program was coded onto computers and this software is relatively new, there could have been existing errors that impacted the calculation of the RMSD values.

To improve this reliability, the SeeSAR data files could be inputted into other software that is more reliable to calculate a better RMSD value, aforementioned above. As SeeSAR does not produce RMSD value, the change in what program to use to calculate the RMSD value can greatly impact future results. Since the DiffDock-Pocket program is relatively new, there are still improvements that can be made to the code. This could include updating the software to stay in line with new advancements in science regarding protein-ligand binding. Since DiffDock-Pocket is a machine learning program, to increase reliability, more proteins can be introduced into the program so that the program learns the patterns of docking and is familiarized with multiple programs.

Additionally, the processing unit that the SeeSAR and DiffDock-Pocket software were run on can affect the overall result and interpretation of the program. If this software were run on a graphics processing unit (GPU) instead of a central processing unit (CPU), there would be improvements in the data collection and overall interpretation of the general software. GPUs have better compatibility with the software, and have a better running time, increasing the amount of proteins that can be processed at one time. This consideration can be used to identify the best conditions that can be set up to produce the best results regarding the protein docking software.

Further Research

Some areas of improvement or possibilities of further research include more of a focus on hybrid approaches. Developing hybrid methods that combine the strengths of traditional and AI-based approaches can be used to improve the accuracy and efficiency of the process as a whole. Traditional molecular docking approaches may be effective at providing the frameworks for molecular interactions, but they are limited in the ability to capture the full complexity of molecular behavior in particularly complex systems or compounds. Machine docking that also holds the ability to predict outcomes from traditional approaches can lead to more innovations in the field.

AI-based methods can synthesize information and patterns from a vast array of data, but they are also limited in their specific data fields in terms of reliability. Combining the best of both approaches would be something to look further into, and the combination of traditional and AI-based methods would enhance the capabilities of docking tools.

Improving the quality and diversity of training data is another aspect for further research, as large, well-curated datasets are essential for training AI models. Reducing bias and enhancing different models' abilities to generalize in creating results for more expansive data would lead to a significant boost in performance and reliability.

Another area of improvement is to include statistical analysis to compare the RMSD results for both the SeeSAR and DiffDock-Pocket. 135 proteins were used to test the SeeSAR and the DiffDock-Pocket programs, which is not a statistically significant number from the full PDB database to create an accurate statistical interpretation of the data. A T-test would not be appropriate for this dataset, which limits the analysis of the results due to the lack of determination of a statistical difference between the RMSDs calculated by both programs. For future studies, a large enough dataset should be used with a sufficient amount of time to be able to apply a T-test statistical analysis, helping further the conclusive results of the efficiency and correctness of the two programs. Notably, as DockGen only contains 189 protein-ligand combinations, they should also seek to expand this dataset while maintaining its generalization-testing characteristics.

Furthermore, using a variety of RMSD calculators may reduce errors that could occur due to bugs or other coding errors. Ensuring that the RMSDs are calculated correctly, especially how it takes into account atom misalignment, is imperative for future studies that use this metric to compare the accuracy of traditional and machine learning docking programs.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. Hannes Stärk and Mr. Michael Plainer for their support throughout the process and for teaching us about DiffDock and DiffDock-Pocket. We would like to thank Drew University for providing the facilities used during the Governor's School of New Jersey program in the Sciences. Thank you to Dr. Cincotta for his guidance during our project and for contributing to our daily mindset and real-world experience. Thank you to Ian Weitz, Visvam Rajest, Kaleb Yong, and Yeji Kim for assistance with data collection. We appreciate Katerina Pouathas and Joel Moses for their coordination and assistance with the project as a whole. Finally, thank you to the Overdeck Family Foundation, Novartis Pharmaceuticals, and the New Jersey Office of the Secretary of Higher Education for the essential funding that has offered the GSNJS to thousands of students across the past 41 years. Without their help, there would be no research to begin with, so once again, thank you to the Overdeck Family Foundation, Novartis, and the New Jersey Office of the Secretary of Higher Education for supporting all the scholars' passion for science.

REFERENCES

Abagyan, R.; Kufareva, I. Methods of Protein Structure Composition. National Institute of Health Methods Mol Biol. 2012; 857: 231–257.https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4321859/ (accessed 2024-07-15).

Balius, T.; Rigby, M. Molecular Docking and Machine Learning - NCI. National Cancer Institute. https://www.cancer.gov/research/key-initiatives/ras/news-events/dialogue-blog/2022/dock (accessed 2024-07-15).

Ballante, F.; Kooistra, A. J.; Kampen, S.; de Graaf, C.; Carlsson, J. Structure-Based Virtual Screening for Ligands of G Protein–Coupled Receptors: What Can Molecular Docking Do for You? Pharmacological Reviews 2021, 73 (4), 527–565. https://doi.org/10.1124/pharmrev.120.000246 (accessed 2024-07-15)..

Bankaitis, V.; Tripathi, A. Molecular Docking: From Lock and Key to Combination Lock. National Institute of Health 2017, 2(1):10.16966/2575-0305.106. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5764188/ (accessed 2024-07-15).

Campbell Biology 12th Ed. (2020) - L. Urry et al. (accessed 2024-07-15).

Clyde, A.; Liu, X.; Brettin, T.; Yoo, H.; Partin, A.; Yadu Babuji; Blaiszik, B.; Jamaludin Mohd-Yusof; Merzky, A.; Turilli, M.; Jha, S.; Ramanathan, A.; Stevens, R. AI-Accelerated Protein-Ligand Docking for SARS-CoV-2 Is 100-Fold Faster with No Significant Change in Detection. Scientific Reports 2023, 13 (1). https://doi.org/10.1038/s41598-023-28785-9 (accessed 2024-07-15).

Corso, G.; Deng, A.; Fry, B.; Polizzi, N.; Barzilay, R.; Jaakkola, T. Published as a Conference Paper at ICLR 2024 Deep Confident Steps to New Pockets: Strategies for Docking Generalization; 2024. https://arxiv.org/pdf/2402.18396 (accessed 2024-07-15).

D. S. Goodsell; G. M. Morris; A. J. Olson. Automated docking of flexible ligands: applications of AutoDock. Journal of molecular recognition 1996, 9(1), 1–5. https://doi.org/10.1002/(sici)1099-1352(199601)9:1<1::aid-jmr241>3.0.co;2-6

Dar, A. M.; Mir, S. Molecular Docking: Approaches, Types, Applications and Basic Challenges. Journal of Analytical & Bioanalytical Techniques 2017, 08 (02). https://doi.org/10.4172/2155-9872.1000356 (accessed 2024-07-15).

Fan, J.; Fu, A.; Zhang, L. Progress in Molecular Docking. Quantitative Biology 2019, 7 (2), 83–89. https://doi.org/10.1007/s40484-019-0172-y (accessed 2024-07-15).

Ferreira, L.; dos Santos, R.; Oliva, G.; Andricopulo, A. Molecular Docking and Structure-Based Drug Design Strategies. Molecules 2015, 20 (7), 13384–13421. https://doi.org/10.3390/molecules200713384 (accessed 2024-07-15).

G. Jones; P. Willett; R. C. Glen; A. R. Leach; R. Taylor. Development and validation of a genetic algorithm for flexible docking. Journal of molecular biology 1997, 267(3), 727–748. https://doi.org/10.1006/jmbi.1996.0897 (accessed 2024-07-15).

Gao, K.; Pei, Q.; Zhu, J.; He, K.; Wu, L. FABind+: Enhancing Molecular Docking through Improved Pocket Prediction and Pose Generation; 2024. https://arxiv.org/pdf/2403.20261 (accessed 2024-07-15).

Gentile, F.; Yaacoub, J. C.; Gleave, J.; Fernandez, M.; Ton, A.-T.; Ban, F.; Stern, A.; Cherkasov, A. Artificial Intelligence–Enabled Virtual Screening of Ultra-Large Chemical Libraries with Deep Docking. Nature Protocols 2022, 17 (3), 672–697. https://doi.org/10.1038/s41596-021-00659-2 (accessed 2024-07-15).

Han, R.; Yoon, H.; Kim, G.; Lee, H.; Lee, Y. Revolutionizing Medicinal Chemistry: The Application of Artificial Intelligence (AI) in Early Drug Discovery. Pharmaceuticals 2023, 16 (9), 1259. https://doi.org/10.3390/ph16091259 (accessed 2024-07-15).

Ketata, M.; Laue, C.; Mammadov, R.; Stärk, H.; Wu, M.; Corso, G.; Marquet, C.; Barzilay, R.; Jaakkola, T. DiffDock-PP: Rigid Protein-Protein Docking with Diffusion Models; 2023. https://arxiv.org/pdf/2304.03889 (accessed 2024-07-15).

Lehninger Principles of Biochemistry 8th Ed. (2021) - D. Nelson et al. (accessed 2024-07-15).

Libouban, P. The Impact of Data on Structure-Based Binding Affinity Predictions Using Deep Neural Networks. National Institute of Health 2023, 24(22): 16120.. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10671244/ (accessed 2024-07-15).

Liu, M.; Wang, S. McDock: a Monte Carlo simulation approach to the molecular docking problem. Journal of computer-aided molecular design 1999, 13(5), 435–451. https://doi.org/10.1023/a:1008005918983 (accessed 2024-07-15).

Liu, Z.; Su, M.; Han, L.; Liu, J.; Yang, Q.; Li, L.; Wang, R. Forging the basis for developing protein–ligand interaction scoring functions. Accounts of Chemical Research, 50, 2017. https://pubs.acs.org/doi/full/10.1021/acs.accounts.6b00491 (accessed 2024-07-15).

Masters, M. R.; Mahmoud, A. H.; Wei, Y.; Lill, M. A. Deep Learning Model for Efficient Protein–Ligand Docking with Implicit Side-Chain Flexibility. Journal of chemical information and modeling 2023, 63 (6), 1695–1707. https://doi.org/10.1021/acs.jcim.2c01436 (accessed 2024-07-15).

McNutt, A. T.; Francoeur, P.; Aggarwal, R.; Masuda, T.; Meli, R.; Ragoza, M.; Sunseri, J.; Koes, D. R. GNINA 1.0: Molecular Docking with Deep Learning. Journal of Cheminformatics 2021, 13 (1). https://doi.org/10.1186/s13321-021-00522-2 (accessed 2024-07-15).

Pei, Q. NeurIPS Poster FABind: Fast and Accurate Protein-Ligand Binding. neurips.cc. https://neurips.cc/virtual/2023/poster/71739# (accessed 2024-07-24).

Peter Chinedu Agu; Celestine Azubuike Afiukwa; Orji, O. U.; Ezeh, E. M.; Ofoke, I. H.; Ogbu, C. O.; Emmanuel Ike Ugwuja; Aja, P. M. Molecular Docking as a Tool for the Discovery of Molecular Targets of Nutraceuticals in Disease Management. Scientific Reports 2023, 13 (1). https://doi.org/10.1038/s41598-023-40160-2 (accessed 2024-07-15).

Pinzi, L.; Rastelli, G. Molecular Docking: Shifting Paradigms in Drug Discovery. International Journal of Molecular Sciences 2019, 20 (18), 4331. https://doi.org/10.3390/ijms20184331 (accessed 2024-07-15).

Plainer, M.; Toth, M.; Dobers, S.; Stärk, H.; Corso, G.; Marquet, C.; Barzilay, R. https://plainer.dev/assets/pdf/2023/DiffDock-Pocket.pdf

Qing, R.; Hao, S.; Smorodina, E.; Jin, D.; Zalevsky, A.; Zhang, S. Protein Design: From the Aspect of Water Solubility and Stability. National Institute of Health 2022, 122(18): 14085–14179. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9523718/ (accessed 2024-07-15).

Sharma, A.; Kunwar, S.; Agarwal, V.; Singh, C.; Sharma, M.; Chauhan, N. Molecular Docking: An Explanatory Approach Structure-Based Drug Designing and Discovery, International Journal of Pharmacy and Pharmaceutical Sciences, 2021. https://www.researchgate.net/publication/352094390_MOLECULAR_DOCKING_AN_EXPLANATORY_APPROACH_IN_STRUCTURE-BASED_DRUG_DESIGNING_AND_DISCOVERY (accessed 2024-07-15).

Torres, P.; Sodero, A.; Jofily, P; Silva-Jr, F. Key Topics in Molecular Docking for Drug Design. National Institute of Health 2019, 20(18): 4574. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6769580/ (accessed 2024-07-15).

Wikipedia Contributors. Docking (molecular). Wikipedia. https://en.wikipedia.org/wiki/Docking_%28molecular%29 (accessed 2024-07-15).

Xu, X.; Huang, M.; Zou, X. Docking-based inverse virtual screening: methods, applications, and challenges. National Institute of Health 2018, 4(1): 1–16. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5860130/ (accessed 2024-07-15).

Yu, Y.; Lu, S.; Gao, Z.; Zheng, H.; Ke, G. Do Deep Learning Models Really Outperform Traditional Approaches in Molecular Docking?; 2023. https://arxiv.org/pdf/2302.07134 (accessed 2024-07-15).

Zhang, X.; Shen, C.; Zhang, H.; Kang, Y.; Hsieh, C.-Y.; Hou, T. Advancing Ligand Docking through Deep Learning: Challenges and Prospects in Virtual Screening. Accounts of chemical research 2024, 57. https://doi.org/10.1021/acs.accounts.4c00093 (accessed 2024-07-15).

Zhang, Y.; Bell, E. W. DockRMSD: Docking Pose Distance Calculation. zhanggroup.org. https://zhanggroup.org/DockRMSD/ (accessed 2024-07-15).

APPENDIX A: PYTHON SCRIPTS FOR CALCULATING RMSD VALUES

A-1: Script for DiffDock-Pocket

```
import os
import numpy as np
from rdkit import Chem
from rdkit import RDLogger

RDLogger.DisableLog('rdApp.*')

def load_mol_from_sdf(file_path):
  if not os.path.exists(file_path):
    raise FileNotFoundError(f"File not found: {file_path}")

  suppl = Chem.SDMolSupplier(file_path)
  mol = next(suppl)
  if mol is None:
    raise ValueError(f"Could not load molecule from SDF file: {file_path}")
  return mol

def load_mol_from_mol2(file_path):
  if not os.path.exists(file_path):
    raise FileNotFoundError(f"File not found: {file_path}")

  mol = Chem.MolFromMol2File(file_path)
  if mol is None:
    raise ValueError(f"Could not load molecule from MOL2 file: {file_path}")
  return mol
```

```
def get_atoms_from_mol(mol, remove_hydrogens=True):
  conf = mol.GetConformer()
  atoms = np.array([conf.GetAtomPosition(i) for i in range(mol.GetNumAtoms())])
  if remove_hydrogens:
    heavy_atoms = [atom.GetIdx() for atom in mol.GetAtoms() if atom.GetAtomicNum() >
1]
    atoms = atoms[heavy_atoms]
  return atoms

def compute_RMSD(atoms1, atoms2):
  if len(atoms1) != len(atoms2):
    raise ValueError(f"Atom sets must have the same number of atoms. atoms1:
{len(atoms1)}, atoms2: {len(atoms2)}")
  centroid1 = np.mean(atoms1, axis=0)
  centroid2 = np.mean(atoms2, axis=0)
  atoms1 -= centroid1
  atoms2 -= centroid2
  return np.sqrt(np.mean(np.sum((atoms1 - atoms2) ** 2, axis=1)))

def get_atom_symbols(mol):
  return [atom.GetSymbol() for atom in mol.GetAtoms()]

def compare_atom_order(output_file, truth_file):
  output_mol = load_mol_from_sdf(output_file)
  truth_mol = load_mol_from_mol2(truth_file)

  output_atoms = get_atom_symbols(output_mol)
  truth_atoms = get_atom_symbols(truth_mol)

  return output_atoms == truth_atoms
```

```
def main():
  path = '' #insert path
  output_file = os.path.join(path, 'results.txt')

  with open(output_file, 'w') as f:
     for file in os.listdir(path):
        if file.endswith('_output.sdf'):
           protein_name = file.replace('_output.sdf', '')
           print(protein_name)
           diffdock_output = os.path.join(path, f'{protein_name}_output.sdf')
           ground_truth = os.path.join(path, f'{protein_name}_truth.mol2')

           try:
              if not compare_atom_order(diffdock_output, ground_truth):
                 raise ValueError(f"Atom orders in {protein_name}_output.sdf and
{protein_name}_truth.mol2 do not match.")

              rank1_mol = load_mol_from_sdf(diffdock_output)
              ground_truth_mol = load_mol_from_mol2(ground_truth)

              atoms1 = get_atoms_from_mol(rank1_mol)
              atoms2 = get_atoms_from_mol(ground_truth_mol)

              if len(atoms1) != len(atoms2):
                 f.write(f"Number of atoms in {protein_name}_output.sdf: {len(atoms1)}\n")
                 f.write(f"Number of atoms in {protein_name}_truth.mol2: {len(atoms2)}\n")

              rmsd = compute_RMSD(atoms1, atoms2)

              f.write(f"RMSD for {protein_name}: {rmsd}\n")
              if rmsd < 2:
```

```
                f.write("Docking is considered successful.\n\n")
            else:
                f.write("Docking is not considered successful.\n\n")
        except Exception as e:
            f.write(f"An error occurred for {protein_name}: {e}\n")
        f.write("\n")

if __name__ == "__main__":
  main()
```

A-2: Script for SeeSAR

```
import os
import numpy as np
from rdkit import Chem
from rdkit import RDLogger
import glob

RDLogger.DisableLog('rdApp.*')

def load_mol_from_sdf(file_path):
  if not os.path.exists(file_path):
    raise FileNotFoundError(f"File not found: {file_path}")

  suppl = Chem.SDMolSupplier(file_path)
  mol = next(suppl)
  if mol is None:
    raise ValueError(f"Could not load molecule from SDF file: {file_path}")
  return mol

def load_mol_from_mol2(file_path):
```

```
  if not os.path.exists(file_path):
      raise FileNotFoundError(f"File not found: {file_path}")

  mol = Chem.MolFromMol2File(file_path)
  if mol is None:
      raise ValueError(f"Could not load molecule from MOL2 file: {file_path}")
  return mol

def remove_hydrogens(mol):
  return Chem.RemoveHs(mol)

def get_atoms_from_mol(mol):
  conf = mol.GetConformer()
  atoms = np.array([list(conf.GetAtomPosition(i)) for i in range(mol.GetNumAtoms())])
  return atoms

def compute_RMSD(atoms1, atoms2):
  if len(atoms1) != len(atoms2):
      raise ValueError(f"Atom sets must have the same number of atoms. atoms1:
{len(atoms1)}, atoms2: {len(atoms2)}")
  centroid1 = np.mean(atoms1, axis=0)
  centroid2 = np.mean(atoms2, axis=0)
  atoms1 -= centroid1
  atoms2 -= centroid2
  return np.sqrt(np.mean(np.sum((atoms1 - atoms2) ** 2, axis=1)))

def get_atom_info(mol):
  conf = mol.GetConformer()
  return [(atom.GetSymbol(), atom.GetAtomicNum(),
conf.GetAtomPosition(atom.GetIdx())) for atom in mol.GetAtoms()]
```

```
def sort_atoms(atom_info):
  return sorted(atom_info, key=lambda x: (x[1], x[2].x, x[2].y, x[2].z))

def match_atom_order(mol, ref_mol):
  ref_atom_info = get_atom_info(ref_mol)
  ref_atoms_sorted = sort_atoms(ref_atom_info)

  mol_atom_info = get_atom_info(mol)
  mol_atoms_sorted = sort_atoms(mol_atom_info)
  ref_to_mol_index_map = {ref_atom_info.index(atom):
mol_atom_info.index(sorted_atom) for atom, sorted_atom in zip(ref_atoms_sorted,
mol_atoms_sorted)}

  new_order = [ref_to_mol_index_map[i] for i in range(len(ref_atom_info))]
  mol = Chem.RenumberAtoms(mol, new_order)
  return mol

def compare_atom_order(output_file, truth_file):
  output_mol = load_mol_from_sdf(output_file)
  truth_mol = load_mol_from_mol2(truth_file)

  output_mol = remove_hydrogens(output_mol)
  truth_mol = remove_hydrogens(truth_mol)

  output_mol = match_atom_order(output_mol, truth_mol)

  output_atoms = sort_atoms(get_atom_info(output_mol))
  truth_atoms = sort_atoms(get_atom_info(truth_mol))

  output_symbols = [atom[0] for atom in output_atoms]
  truth_symbols = [atom[0] for atom in truth_atoms]
```

```
  if output_symbols != truth_symbols:
      print(f"Output atoms: {output_symbols}")
      print(f"Truth atoms: {truth_symbols}")
      return False
  return True

def main():
  path = os.getcwd()
  output_file = os.path.join(path, 'results.txt')

  with open(output_file, 'w') as f:
      for file in os.listdir(path):
          if file.endswith('.sdf'):
              protein_name = file[:4]
              print(protein_name)

              files = glob.glob(f'*{protein_name}*.sdf')
              file_to_open = files[0]

              new_files = glob.glob(f'*{protein_name}*.mol2')
              new_files_2 = new_files[0]

              seesar_output = os.path.join(path, file_to_open)
              ground_truth = os.path.join(path, new_files_2)

              try:
                  if not compare_atom_order(seesar_output, ground_truth):
                      raise ValueError(f"Atom orders in {protein_name}_output.sdf and
{protein_name}_truth.mol2 do not match.")
```

```
            rank1_mol = load_mol_from_sdf(seesar_output)
            ground_truth_mol = load_mol_from_mol2(ground_truth)

            rank1_mol = remove_hydrogens(rank1_mol)
            ground_truth_mol = remove_hydrogens(ground_truth_mol)

            atoms1 = sort_atoms(get_atom_info(rank1_mol))
            atoms2 = sort_atoms(get_atom_info(ground_truth_mol))

            atoms1_coords = np.array([atom[2] for atom in atoms1])
            atoms2_coords = np.array([atom[2] for atom in atoms2])

            if len(atoms1_coords) != len(atoms2_coords):
                f.write(f"Number of atoms in {protein_name}_output.sdf:
{len(atoms1_coords)}\n")
                f.write(f"Number of atoms in {protein_name}_truth.mol2:
{len(atoms2_coords)}\n")

            rmsd = compute_RMSD(atoms1_coords, atoms2_coords)

            f.write(f"RMSD for {protein_name}: {rmsd}\n")
            if rmsd < 2:
                f.write("Docking is considered successful.\n\n")
            else:
                f.write("Docking is not considered successful.\n\n")
        except Exception as e:
            f.write(f"An error occurred for {protein_name}: {e}\n")
        f.write("\n")

if __name__ == "__main__":
  main()
```

APPENDIX B: PROTEIN RMSD VALUES

Protein	Ligand	SeeSAR RMSD Values	DiffDock-Pocket RMSD Values
6YAQ	OHZ	6.548556415	3.969967932
6YAP	OHZ	5.642864082	5.306365366
6YAO	OJ2	4.762559454	5.491974076
6XUG	O1Q	NR	NR
6XB3	9BG	7.059221742	3.236732115
6WYZ	DGL	1.344460215	4.460617031
6TVG	AP2	6.950241341	NR
6RZ2	5CD	0.3610490441	5.060804164
6RYZ	SAM	4.924733776	4.82293686
6RMS	AMP	4.809970876	2.087711926
6QKR	FAD	NR	2.796469307
6PAA	ASP	1.120114167	3.272102996
6PA6	ASN	2.612944753	3.146493673
6PA2	ASP	2.302639679	2.690919323
6O6Y	ACK	4.619473552	4.093633237
6NPP	KWG	NR	0.6233687372
6NCO	KQP	2.734716762	3.763479099
6JLS	FMN	NR	3.811641111
6GBF	AMP	4.790230224	1.890267314
6FGC	D95	NR	NR
6FGC	ADP	NR	NR
6EP5	ADP	0.6931154064	3.264701234
6EA9	9BG	4.05350537	3.463187062
5MH1	BMA	NR	1.166916714
5IDA	BMA	0.2590456832	0.1492579507
5HQX	EDZ	1.996665655	5.627132371
5HMR	FDZ	NR	NR
5HHZ	ZME	3.346694758	3.804462098
5GQL	ATP	NR	NR
5GQI	ATP	NR	4.928089755

5FXF	BEZ	1.658471337	2.206931921
5FXE	CIY	3.800775292	5.192293684
5FXD	H7Y	2.557879122	3.495396633
5FIU	Y3J	3.662642351	5.605927519
5F2T	PLM	4.534741314	2.24896224
5ERS	AMP	NR	3.738123977
5ENT	MIY	4.731718745	4.338300175
5ENR	MBX	3.944619418	4.710783474
5ENQ	5QE	6.223384263	4.943821234
5ENP	5QF	8.216089624	8.277075295
5ENO	5QG	3.782946369	7.170207856
5B5S	BOG	4.559091268	8.249798342
5AE3	AWB	NR	NR
5A98	ATP	NR	6.07128208
4ZQX	ATP	3.830513761	3.285957777
4ZAZ	4LS	NR	7.248102885
4ZAY	4LS	NR	7.749953996
4ZAW	4LU	NR	6.711632241
4YDX	TCE	4.274894818	4.700182334
4XDR	ADN	NR	NR
4UUW	AMP	32.67616725	NR
4UOC	NCN	4.222454525	6.378105246
4U63	FAD	NR	8.795904725
4TVD	BGC	2.50684279	NR
4TVD	GLC	2.346946136	NR
4RPM	HXC	NR	5.454164856
4RPJ	UDP	2.733882387	3.636867288
4RHE	FMN	2.602152337	0.7308548718
4QO5	NAG	NR	4.212264736
4QA8	PJZ	NR	8.77685151
4PYA	2X3	4.945816973	6.374328476
4PHS	UDP	2.516440351	6.467255689
4PHR	UDP	5.24269893	5.188844522

4PFX	UDP	5.140841944	3.670383255
4OAL	245	4.023719415	2.289794836
4O95	245	3.891900355	0.6851603741
4N4L	HG1	NR	3.380265073
4MO2	FDA	9.69716604	2.276164317
4MIG	G3F	2.711243058	2.938644821
4KGX	CTP	5.083507815	1.920712949
4IDK	1FE	2.478426626	5.587841531
4H2F	ADN	4.365733579	2.201964331
4FYV	DCP	3.548969753	4.431817167
4CDN	FAD	NR	6.512035489
4B4V	L34	4.245906557	1.193430768
3ZZS	TRP	2.769142831	2.564446549
3ZQU	FNR	0.165267156	1.445643545
3ZJX	BOG	8.03904357	7.897796784
3ZEC	ANP	6.186063048	3.418762469
3WVC	FEG	4.834995597	9.462925346
3WRB	GDE	3.267663543	3.967564696
3UNI	SAL	0.2316973675	NR
3SR6	MTE	2.107852352	1.2002487
3S6A	ANP	NR	2.338078985
3O7J	2AL	1.893570713	2.484294936
3O02	JN3	3.384768983	3.73995299
3O01	DXC	0.1880231854	2.656015557
3NVV	MTE	2.400064658	1.010109711
3JU4	SLB	3.680213607	2.899652892
3JQM	GTP	6.330146036	5.899657114
3INR	GDU	2.843892142	8.06395327
3HE3	UDP	1.815152614	4.753077278
3GVL	SLB	3.746764094	5.123826546
3GF4	FAD	NR	NR
3GF4	U5P	3.194132609	4.325258282
3ADA	NAD	NR	NR

3AD9	NAD	4.824605225	2.378809369
3AD7	NAD	4.825967271	2.710316689
2ZE9	CTP	NR	NR
2ZD0	TRP	0.2021660765	3.573199721
2ZCZ	TRP	2.643512844	2.60169933
2XTA	ACO	NR	5.368482035
2X34	UQ8	22.94504968	11.13828582
2WR8	SAH	2.814870481	6.910348084
2VFU	MTL	2.819392031	2.171795649
2V7W	5FD	4.685957752	0.8078816797
2V7V	5FD	4.054003684	1.275065296
2V7U	SAM	7.341500083	1.434025668
2V7T	SAH	5.261254791	1.142981512
2R4E	13P	71.61439858	3.859629243
2Q6K	ADN	2.765687302	6.246294716
2Q37	3AL	2.895969186	4.220689483
2O5M	MNR	NR	NR
2HS3	FGR	5.52945136	6.147321023
2HK9	SKM	4.457953205	4.452990084
2GF3	FOA	1.737074784	1.667811227
2GAH	NAD	3.959753928	1.916679078
2GAG	NAD	5.043834382	2.510647284
2EXT	TRP	3.052264616	2.608067231
2CDC	XYS	2.201415376	2.633765335
1ZA2	CTP	6.037665672	3.182398381
1V97	MTE	NR	NR
1UF8	ING	3.412926135	2.761607771
1UF7	CDV	0.9448023875	3.717648487
1UF5	CDT	4.501598385	3.267865869
1TKG	SSA	4.953213626	7.614046095
1SIJ	PCD	NR	NR
1SBZ	FMN	6.372276729	7.884041137
1RQP	SAM	1.20138935	3.385661311

1QAW	TRP	1.413155178	2.583685965
1PJ4	FUM	1.960481709	2.679683276
1PJ2	FUM	2.880565485	3.62803934
1O72	PC	3.088491695	5.045060787
1O28	UFP	1.55518133	5.339365103
1I8	FAD	3.968682227	3.602765153