Optimization of Protein-Ligand Molecular Docking using Graph Convolutional Neural Networks: Predicting Binding Affinity and posing

A Graduate Project Report

Submitted to

San José State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Engineering

By

Yolanda Marquez

Yang Liu

Supriya Kankati

Akash Jagarlamudi

FALL 2024

Preface

As part of the Master in Engineering curriculum we have created a report on the topic "Optimization of Protein-Ligand Molecular Docking Using Graph-Convolutional Neural Networks: Predicting Binding Affinity. The main objective of this project was to to develop a Graph Convolutional Neural Network (GCN) - based Framework to directly predict binding affinity and pose evaluation from several protein-ligand complexes. We aim to address the limitations of traditional docking tools by introducing this machine learning based approach that learns molecular interaction patterns from data. The basis of this project originally stemmed from our interest in pharmaceutical drug applications with the current technologies they use today and incorporating machine learning to integrate to a single workflow. As of 2022, most methods have not been proposed and assessed in a complete docking pipeline where classical methods such as Autodock4 (AD4) and Autodock Vina (Vina) are compared to a machine learning workflow, so we aim to bridge this gap [21]. The areas we are concentrating on are drug discovery optimization, small molecule ligands, deep learning, machine learning, and artificial intelligence.

Acknowledgements

We extend our heartfelt gratitude to our graduate mentor and advisor, Dr. Ahmed Hambaba, for his invaluable guidance and support throughout this project. As the primary advisor, his insightful foundation for ML/AI approaches and biweekly check-ins to monitor our progress were instrumental to our success. We also sincerely thank Dr. Anand Ramasubramanian of the Chemical and Materials Engineering Department for his foundational guidance in developing the initial project idea and expertise in topics such as Autodock and Vina traditional docking approaches. The integration of the docking foundational work with Dr. Hambaba's ML/AI insights was crucial to shaping this project and ensuring its successful execution.

Additionally, we appreciate the SJSU Engineering IT department for granting us access to High-Performance Computing (HPC) resources, and we are thankful to Dr. Ramasubramanian for facilitating access to his university lab, which enabled the use of these HPC systems.

We are deeply grateful for these resources and for the unwavering support of our professors, colleagues, friends and families, who have been instrumental in the success of this project. Thank you.

The Designated Advisor Approves the Report Titled

Optimization of Protein-Ligand Molecular Docking Using Graph Convolutional Neural Networks: Predicting Binding Affinity

By

Yolanda Marquez

Yang Liu

Supriya Kankati

Akash Jagarlamudi

APPROVED FOR THE COURSE OF ENGR-295B Fall 2024 Master's Project

DEPARTMENT OF MASTER OF SCIENCE IN ENGINEERING

SAN JOSÉ STATE UNIVERSITY

Fall 2024

PRIMARY ADVISOR: Prof. Ahmed Hambaba **Date**

COMMITTEE MEMBERS: **Date**

COURSE INSTRUCTOR: Prof. Ahmed Hambaba **Date**

Table of Contents

Abstract

Molecular docking is an in-silico method widely utilized in early-stage drug discovery for screening promising drug candidates and exploring potential side effects or toxicities [8]. Traditionally, tools like AutoDock4 (AD4) and AutoDock Vina (Vina) estimate protein-ligand binding affinities using heuristic scoring functions, providing a balance between computational efficiency and accuracy. However, these methods face challenges such as limited ability to capture nuanced molecular interactions, rigid receptor assumptions, and inaccuracies in pose prediction, particularly in complex or flexible systems. These limitations significantly contribute to the inefficiencies observed in drug discovery, where only 20-30% of compounds identified through molecular docking show activity in biological assays [22]. While AD4 and Vina remain valuable for initial screenings, their results often require refinement to align with experimental outcomes.

To address and overcome these challenges, we propose a novel framework integrating Graph Convolutional Neural Networks (GCNs) with traditional docking software. GCNs excel in modeling protein-ligand complexes as graph-based data, enabling more precise predictions of binding affinities and identifying the most energetically favorable configurations. This method enhances the predictive accuracy and computational efficiency of docking processes. By combining the output of traditional tools like AD4 and Vina with the predictive capabilities of GCNs, our framework aims to optimize binding affinity prediction and pose identification [1]. This integrated approach leverages machine learning to learn from molecular docking patterns, significantly enhancing both accuracy and efficiency in computational drug discovery, potentially reducing the time and costs associated with developing new therapeutics.

Technical Objectives

The primary objective of this project is to develop a machine learning framework based on Graph Convolutional Networks (GCNs) to improve the accuracy and efficiency of molecular docking in the drug discovery process. The specific objectives of this project include:

1. **Develop an optmized traditional docking procedure** to pre-process file outputs.
2. **Designing a GCN model** that accurately predicts the binding affinities of protein-ligand complexes.
3. **Integrating GCNs with traditional molecular docking tools** (e.g., AutoDock, Vina) to enhance docking predictions and reduce computational time.
4. **Evaluating the performance** of the proposed GCN-based framework by comparing it to traditional molecular docking methods in terms of accuracy, speed, and scalability.
5. **Optimizing the framework** to handle large datasets and diverse protein-ligand complexes, ensuring robustness across different therapeutic areas. This approach will streamline the drug discovery stage pipeline, reducing time and cost while improving the accuracy and reliability of predictions.

Introduction

The process of drug discovery is complex, time-consuming, and costly. Traditional molecular docking techniques, such as AutoDock and Vina, are essential tools in identifying potential drug candidates by simulating interactions between proteins and ligands. However, these methods often fall short in accurately predicting binding affinities and poses due to their reliance on heuristic scoring functions and predefined rules. The limitations of these traditional methods, combined with the high computational costs associated with large datasets, hinder the

efficiency of the drug discovery process. To address these challenges, we propose a novel approach that combines traditional docking methods with machine learning, specifically Graph Convolutional Networks (GCNs), to improve prediction accuracy, speed, and scalability in molecular docking. By leveraging the strengths of GCNs, this framework aims to accelerate the drug discovery process, reducing both time and cost while increasing the reliability of virtual screening.

Significance of the Problem and Opportunity

Since the past decade, computational drug discovery has made significant advancements in being able to identify drugs that target different diseases that successfully get released onto the market. Throughout the life-cycle of drug development, it undergoes 5 phases: discovery and development, preclinical research, clinical research, FDA review and FDA post-market and safety monitoring. The stages of drug development are shown in Figure 1 below.

Figure 1: The 5 phases of Drug Development during its Life-Cycle [35]

In the discovery and development phase, molecular docking is an in-silico technique used to screen hundreds to thousands of drug candidates to identify potential therapeutic agents for further testing through in-vitro and in-vivo experimentation. Extensive virtual screening of candidate molecules leads to a single potential drug for commercialization. Despite its effectiveness, screening vast databases of compounds through traditional docking methods

remains both time-consuming and expensive. As of 2022, the average cost of developing a single drug in the pharmaceutical and biotech industries is approximately $1.8 billion, with the process taking around 12 years to complete [21]. This raises the need to streamline and optimize the drug discovery process to reduce costs and accelerate development.

Traditional docking methods, such as AutoDock and Vina, rely on predefined scoring functions to estimate the strength of protein-ligand interactions. While these methods have their applications, they often struggle to capture the intricate details of molecular interactions, leading to discrepancies when compared to experimental results. Furthermore, these methods suffer from limitations such as high latency, low accuracy, and computational inefficiency, especially when applied to large datasets [16]. The need for more advanced techniques is evident to enhance the accuracy, efficiency, and scalability of molecular docking.

A promising solution to these challenges is the integration of machine learning (ML) models. ML has already shown its potential in leveraging protein mutations to predict binding affinity, thereby improving the accuracy of predictions in the early stages of drug discovery [28]. Moreover, AI-driven drug discovery methods have made significant strides, particularly in accelerating drug repurposing efforts, which became especially important during the COVID-19 pandemic. The ability of AI to rapidly screen existing drugs and identify potential matches for new diseases based on molecular structures and mechanisms has transformed drug discovery [19]. Furthermore, personalized medicine is becoming a reality with AI's ability to predict drug responses tailored to an individual's genetic profile. This not only improves treatment efficacy but also minimizes adverse reactions, marking a significant shift toward more targeted therapies [20].

Given these advancements, our team aims to develop a Graph Convolutional Network (GCN)-based framework, which leverages machine learning to optimize the drug discovery process. GCNs model protein-ligand complexes as graph-based data, capturing the complex molecular interactions and enabling faster and more accurate predictions of binding affinities. This approach offers a scalable and efficient alternative to traditional docking methods, which would significantly reduce the time and costs associated with drug discovery.

Background

Drug discovery is the process of identifying chemical molecules with the potential to become therapeutic agents in medicine through virtual screening. The stages of drug screening are illustrated in Figure 2 below, starting with a protein data bank that can contain thousands to millions of compounds. From there, a few hundred potential drug candidates are selected based on their predicted interactions. Next, in-silico molecular docking methods are used to predict the chemical binding of these molecules through scoring and pose evaluations. Eventually, this will scale down to a few promising candidates that would proceed to in-vivo and in-vitro testing before reaching the commercialization stage.

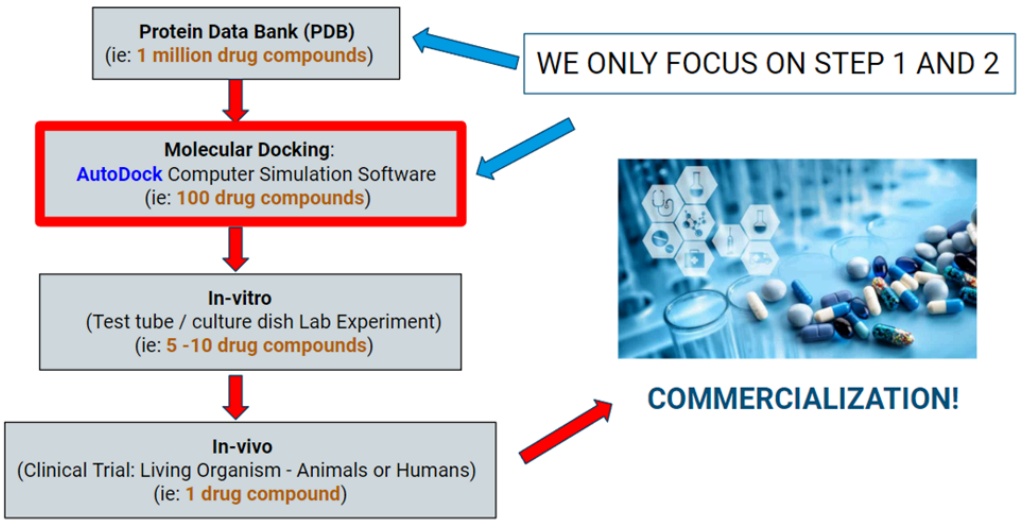

Figure 2: Drug Screening process to commercialization

In the pharmaceutical and biotech industries, the entire life cycle process from drug discovery to commercialization for a single drug based on figure 1 typically takes an average of 12 years and costs approximately $1.8 billion [21]. Moreover, only 20–30% of compounds identified through molecular docking are active in biological testing [22]. This highlights the inefficiency of traditional molecular docking methods and the need for more reliable approaches.

Molecular docking is traditionally performed using software like AutoDock and Vina, which employ heuristic scoring functions to estimate protein-ligand binding affinities. While these methods are useful, they often fall short when it comes to accurately predicting molecular interactions, particularly for complex protein-ligand systems. This is primarily due to their reliance on predefined rules and assumptions, limiting their ability to handle the dynamic nature of molecular interactions. Additionally, these methods are computationally intensive, requiring multiple iterations and simulations, which makes them time- and resource-consuming when applied to large datasets [16].

One of the main challenges faced by traditional docking tools is the pose prediction issue—accurately predicting how a ligand will bind to a protein. Traditional methods also struggle with accounting for the dynamic nature of molecular interactions, which is crucial for accurately predicting binding affinities. Furthermore, these tools often lack generalizability across various types of protein-ligand complexes, making them less effective in addressing the diverse range of biological targets.

To address these limitations, machine learning (ML) techniques have emerged as a promising alternative. ML models are able to consider complex molecular features, including the effects of protein mutations on binding affinity, which can improve the accuracy of predictions.

Moreover, AI in molecular docking has shown considerable promise in the context of drug repurposing, a process that gained prominence during health crises such as the COVID-19 pandemic. AI's ability to quickly screen existing drugs for potential efficacy against new diseases has revolutionized the drug discovery landscape [19]. Additionally, AI-driven molecular docking holds promise for the future of personalized medicine by predicting drug responses based on individual genetic profiles. This approach could lead to more targeted therapies, improving treatment efficacy while minimizing adverse side effects [20].

To enhance the efficiency of drug discovery, our team is focused on developing a Graph Convolutional Network (GCN)-based framework. GCNs represent protein-ligand interactions as graphs, allowing for a more precise understanding of molecular binding. By accurately predicting binding affinities and ranking docking poses, GCNs promise to improve the precision, speed, and scalability of molecular docking, addressing the inefficiencies inherent in traditional methods.

Purpose of Molecular Docking

Discovering bioactive compounds for a given target from a large compound library is one of the major tasks in drug development. It is labor-intensive, time-intensive, and costly to carry out binding affinity measurements on tens or hundreds of thousands of compounds. Therefore, these factors can be greatly reduced with computational methods to effectively predict the binding affinities of compounds before performing the next stage of experiments and eventually, FDA-approved medications [12].

Molecular Docking is an in-silico method in early-stage drug discovery that is used as a

filter to highlight only the most interesting drug candidates and can also detect potential drug side effects and toxicities. It does this by efficiently predicting non-covalent interactions between a target macromolecule protein (receptor) and a small molecule drug candidate (ligand) as shown in *Figure 3* below [12]. It involves Computer-Aided Drug Design (CADD) software that can act as a simulation of what goes on during receptor-ligand complex interactions and find active compounds.

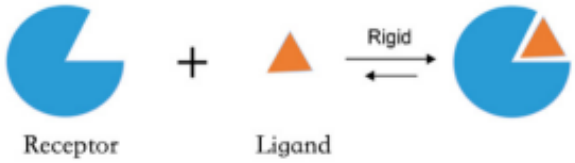

Figure 3: Lock-And-Key Model [5]

Molecular Docking Theory

CADD methods are categorized into 2 types: structure-based drug discovery (SBDD) and ligand-based drug discovery (LBDD). SBDD finds active compounds based on the physical interactions of 3D structures between the target protein and small molecule. LBDD investigates existing activities using quantitative structure-activity relationship (QSAR) models, chemical similarity, pharmacophore, and 3D shape matching to predict the properties of novel compounds. Both SBDD and LBDD can be combined to use for Virtual Screening. For example, scientists can use LBDD to search for ligand compounds similar to available and moderately active compounds from a large library. SBDD can then be used to identify binding sites and interactions to predict protein-ligand interactions to find favorable compounds.

SBDD often uses molecular docking and different programs are available such as *DOCK, AutoDock4, Autodock Vina, GOLD, Glide from Schrödinger, FRED, and Surflex-Dock* as a few examples. All these programs rely on scoring functions (SF) to evaluate protein-ligand binding,

therefore these scores must be robust, quick to calculate, and accurate [12]. For molecular docking, typically 3D structures of 2 molecules are inputted to a program: a target receptor macromolecule (ie: Thrombin) and a ligand small molecule (ie: The drug candidate) as shown in *Figure 4*. Several experimental methods are used to obtain these structures such as *X-ray-based methods* (most prominent), *Nuclear magnetic resonance (NMR)*, and *electron microscopy (EM)*. Fortunately, most structures are already inputted into a Protein Data Bank (PDB) accessible through the Internet as 90% of structures have been solved with *X-ray crystallography* (99% if ligand-protein structures with known binding affinity are taken into account), and 8% have been solved with NMR. Therefore, in this case, the receptor and ligand structure files can be downloaded from PDB and input into a computer software to run molecular docking. When docked together, it mimics the lock-and-key principle model of drug action where the ligand finds and fits into a receptor's optimal binding site with its preferred predicted orientation forming a ligand-receptor stable complex. It does this by sampling first or otherwise generating a set of conformations from a rigid 3D ligand that's evaluated on its capacity to explore a conformational space of a ligand. Secondly, binding affinity or binding energy is evaluated at each receptor-ligand complex formed known as a pose. With virtual screening (VS) or High-throughput virtual screening (HTVS), a comprehensive ligand library can be docked to one target receptor to get binding prediction results such as binding affinity scoring and pose evaluation [21].

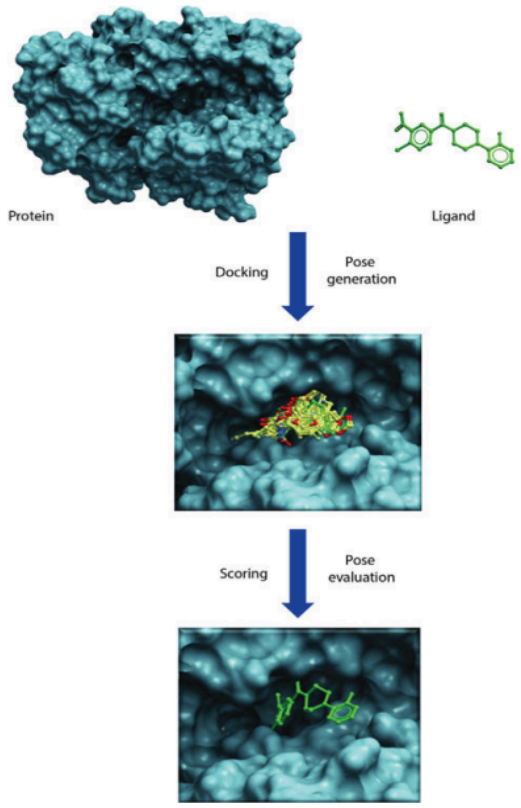

Figure 4: Molecular Docking of Target receptor macromolecule and small molecule ligand drug candidate [5]

Physics-based methods (Autodock 4)

Physics-based scoring functions use molecular physics terms such as Van der Waals interactions (Lennard-Jones potential with 0.5 Å smoothing), electrostatic interactions (coulombic potential), hydrogen bonding, desolvation energies, and torsional entropy (number of a ligand's rotational bonds). These terms can be derived from both experimental data and quantum mechanical calculations [12]. These weighted physics energy terms estimate the free energy aka the final intermolecular energy based on equations 1 and 2 below [3]:

$$\Delta G_{binding} = \mathrm{V} = \Delta G_{vdw} + \Delta G_{elec} + \Delta G_{hbond} + \Delta G_{desolv} + \Delta G_{tors} \qquad \text{(Equation 1)}$$

$$\Delta G_{binding} = W_{vdw}\Sigma_{i,j}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right) + W_{elec}\Sigma_{i,j}\left(\frac{q_i q_j}{\varepsilon(r_{ij}) r_{ij}}\right) + W_{hbond}\Sigma_{i,j}E(t)\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) + W_{sol}\Sigma_{i,j}(S_i V_j + S_j V_i)e^{\left(\frac{-r_{ij}^2}{2\sigma^2}\right)} + W_{tor}N_{tor}$$

(Equation 2)

The inhibition constant (Ki) from Equation 3 is an indication of how effective a ligand acts as an inhibitor to a receptor. Its value (units in nanomoles or nM) represents the concentration required to produce half of the maximum inhibition. In other words, the ligand occupies the receptor's binding site by 50%. The lower the Ki is for a ligand drug, the stronger the binding affinity is for a receptor [37].

$$K_i = e^{\frac{\Delta G}{RT}}$$ (Equation 3)

Where R is the ideal gas constant at 8.314J/K · mol and T is room temperature at 298.15K or 25C.

Autodock4 in particular is a molecular docking program that uses a physics scoring function. It is calibrated with 188 complexes from the Ligand Protein Data bank (LPDB) and has a 2.52 kcal/mol standard error [6]. The limitation to Physics-based scoring functions is that computing the exact value of binding affinity score is computing-intensive, therefore it may run slower, however, it is more accurate compared to the Empirical Scoring Function in its calculations.

Empirical Scoring Functions (Autodock Vina)

Empirical scoring functions estimate the binding affinity of protein-ligand complexes based on a set of weighted scoring terms including VDW, hydrogen bond, hydrophobic, and torsions (proportional to the number of rotatable bonds). Electrostatics and desolvation are not included, unlike the physics-based approach as shown in the equations below. Since these parameters are simpler, it is less computationally intensive. The weights of these descriptors are determined by

fitting experimental binding affinity data of protein-ligand complexes via linear regression [21]. This scoring function draws from both physics-based and knowledge-based and each weight term is learned from training data. Equations 4-6 for empirical scoring functions are shown below [3].

$$\Delta G_{binding} = \mathrm{V} = \Delta G_{vdw} + \Delta G_{hbond} + \Delta G_{hydrophobic} + \Delta G_{tors} \quad \text{(Equation 4)}$$

$$\Delta G_{binding} = \mathrm{V} = (\Delta G_{gauss} + \Delta G_{repulsion}) + \Delta G_{hbond} + \Delta G_{hydrophobic} + \Delta G_{tors} \quad \text{(Equation 5)}$$

$$\Delta G_{binding} = W_{vdw}\Sigma_{i,j}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right) + W_{hbond}\Sigma_{i,j}E(t)\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) + W_{tor}N_{tor} \quad \text{(Equation 6)}$$

Autodock Vina in particular is a molecular docking program that uses an Empirical scoring function. It is calibrated with 1300 complexes from LPDDB and has a 2.85 kcal/mol standard error. This standard error is slightly higher than Autodock 4 given that there are fewer terms used in the equation [6]. An advantage is that this scoring function is less computationally intensive thus results can be achieved quickly. A limitation is that calculating binding affinity is less accurate than a physics-based scoring function. It compensates for this, however, by using the Monte Carlo Iterated Search algorithm combined with Broyden-Fletcher-Goldfarb Shanno (BFGS) for better docking results and better binding poses) [6].

Methodology

This project takes a systematic approach to integrating Graph Convolutional Networks (GCNs) into molecular docking. The process begins with data preparation, where raw molecular structures of ligands and target receptor proteins are cleaned and standardized into consistent formats, such as PDB files to be ready for analysis. Next, docking simulations are performed

using tools like AutoDock4 and Vina to generate initial docking poses—these are potential binding configurations between the ligands and proteins.

In the graph conversion step, molecular structures are transformed into graph representations. Here, atoms are represented as nodes, and their interactions are represented as edges, capturing both the chemical and spatial properties of the molecules. These graphs are then fed into the GCN model for training.

During training, the GCN uses experimental datasets with known binding affinity values as the ground truth. By analyzing these data, the model learns to identify patterns in molecular interactions and continuously refines its parameters to minimize prediction errors. This enables the GCN to predict binding affinities and rank docking poses, effectively identifying the most promising configurations.

By combining traditional docking tools with the advanced capabilities of GCNs, this methodology addresses the shortcomings of conventional approaches, offering a faster, more accurate, and data-driven solution for molecular docking and drug discovery.

Molecular Docking Procedure

In preparation for training our predictive model, the dataset must be meticulously curated and formatted to ensure optimal performance and accuracy. The initial raw data are sourced from the PDBbind v2017 refined set, which comprises 4,154 protein-ligand complex files [16]. To tailor this dataset for use with our docking software, which has specific constraints regarding the complexity and flexibility of ligand structures, we apply stringent filters. Specifically, complexes where the ligand possesses fewer than 5 or more than 32 rotatable bonds are excluded from the dataset. This filtering criterion is predicated on the limitations of the docking software, which is

optimized for ligands within this range of rotatable bonds to ensure computational efficiency and to avoid potential errors in binding pose prediction that can arise from excessive ligand flexibility or rigidity [22]. After manual filtering, the dataset is reduced to 410 protein-ligand complexes, evenly distributed across the entire dataset. Preparing the raw data is time-consuming, with each data file taking approximately 35 minutes to generate. Given the time constraints, only this number of files could be prepared, which is considered a limitation of the current study. This aspect is noted for future improvement in subsequent research efforts. Our dataset can be found at Data_Protein ligand complexes.

The rationale behind setting specific thresholds for rotatable bonds in ligands stems from the need to balance flexibility and computational manageability. Ligands with too few rotatable bonds may not adequately represent the dynamic nature of molecular interactions within the binding site. Conversely, ligands with an excessive number of rotatable bonds could lead to computationally intensive simulations, making the docking process inefficient and potentially less accurate. By filtering out ligands outside the 5 to 32 rotatable bond range, the dataset is optimized to include complexes that are most likely to yield reliable and computationally feasible docking predictions, thereby enhancing the overall robustness and applicability of the model. This preparation limitation is acknowledged as an area for enhancement in future work.

Dataset preparation

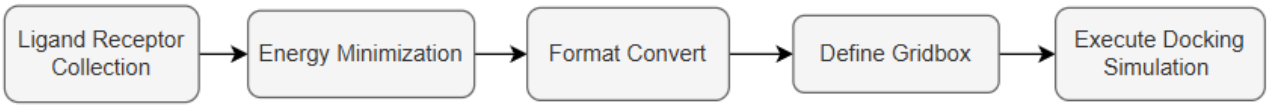

Figure 5 Dataset preparation workflow

Figure 5 illustrates the workflow for the dataset preparation procedure for molecular

docking. The process begins with the collection of ligand-receptor pairs from the PDBbind v2017 refined set, followed by energy minimization to refine the molecular interactions. Next, format conversion is done for compatibility across computational tools, and a grid box is defined to specify the docking search space. Finally, docking simulations are performed to generate the initial poses used for model training. Each step is detailed in the following sections.

Step 1: Collection of Ligand and Receptor Data

In the initial step of preparing the dataset for docking simulations, we focus on the collection and preparation of ligand and receptor data. Traditionally, protein files are utilized; however, for enhanced specificity and efficiency in docking, we employ pocket files instead of full protein structures. Using pocket files allows for concentrating computational resources and analytical accuracy on the regions of the protein directly involved in ligand binding, thus improving docking precision and reducing computational load.

For ligands, initial data are often provided in the Structure-Data File (SDF) format, a widely used format for storing molecular structures and associated data. The SDF format includes structural connectivity information, essential for precise docking simulations, but requires conversion to be compatible with many docking tools.

To make the ligand data compatible with our docking software, we use Discovery Studio, a comprehensive suite of molecular modeling and simulation tools. Discovery Studio facilitates the modification and preparation of ligands by converting the SDF files into Protein Data Bank (PDB) format files. The PDB format is widely used in computational biology for the three-dimensional (3D) representation of protein and ligand structures, making it suitable for use in docking software like AutoDock4 [7], [32].

In this step, we also apply initial filters to the dataset, excluding ligands with fewer than 5 or more than 32 rotatable bonds. This filtering ensures that the ligands included in our study are neither too rigid nor excessively flexible, balancing the accuracy of binding predictions with computational efficiency.

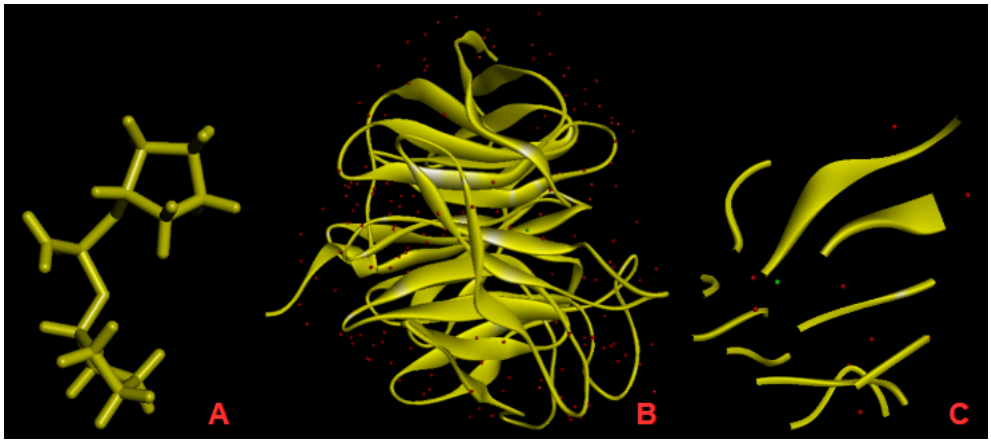

Figure 6: Example of Ligand(A), Protein(B), Pocket Structure(C)

Figure 6 illustrates the general molecular structures involved in this study. *Figure 5* (A) shows the original ligand structure in SDF format, (B) shows the receptor protein structure, and (C) shows the receptor's focused pocket structure. The ligand structure stores the chemical connectivity and 3D orientation information for successful docking. The pocket structure representation emphasizes the targeted area within the protein where the ligand is expected to bind, demonstrating the rationale for using pocket files to focus computational efforts on the most relevant part of the protein [7], [32].

Step 2: Energy Minimization

Following the preparation and conversion of ligands, step two involves the energy minimization of these molecules to ensure they are in their lowest energy conformations before docking. This step is crucial as it directly influences the accuracy of the docking predictions by

simulating more physiologically relevant molecular structures that are likely to occur in natural settings.

For this purpose, we employ Open Babel, an open-source chemical toolbox designed to speak the many languages of chemical data [24]. Open Babel is widely used for converting, generating, analyzing, and manipulating chemical data. It is particularly useful in converting chemical structures from one format to another, preparing files for simulations, and performing tasks such as energy minimization.

The command used for energy minimization is as follows:

>obminimize -o pdb -sd -ff MMFF94 -h -n 50000 ligand.pdb > ligand_EM.pdb

- -o pdb: Specifies the output format as PDB.
- -sd: Indicates the use of the steepest descent algorithm for the initial phase of minimization.
- -ff MMFF94: Selects the Merck Molecular Force Field 94 (MMFF94) as the force field for energy calculations. MMFF94 is renowned for its accuracy in handling a broad range of organic molecules [14], [17].
- -h: Adds hydrogens where necessary, as many ligand structures may not include these in their downloaded or database forms.
- -n 50000: Sets the maximum number of iterations to 50,000, allowing for extensive energy minimization to achieve a stable conformation.

Energy minimization reduces the potential energy of the ligand to a minimum, thereby

stabilizing its conformation. This is important because ligands in lower energy states are more likely to be biologically active and to bind effectively to their target receptors. By ensuring that the ligand is in a minimized energy state, we simulate conditions that are close to the natural physiological environment, enhancing the biological relevance and accuracy of our docking studies [4]. This process also helps in removing any steric clashes or unrealistic bond lengths and angles that might have been introduced during the conversion or manipulation of the ligand structures. This careful preparation sets the stage for more reliable and accurate docking, which is crucial for predicting how these molecules will interact with their target proteins in drug discovery projects.

Step 3: Conversion to PDBQT Format Using MGLTools

The third step is converting the minimized ligand structures into PDBQT format, as it includes additional information such as partial charges and atom types specific to AutoDock's requirements. For this conversion, we use MGLTools, a suite of software provided by the Molecular Graphics Laboratory at The Scripps Research Institute. MGLTools offers graphical user interfaces that facilitate the preparation of input files for AutoDock by converting standard PDB files into the PDBQT format [14]. This software tool not only converts the file format but also assigns AutoDock atom types and calculates Gasteiger charges, to evaluating molecular interactions during docking. For the receptor molecule, kollmann charges have been added so that the software can account for electrostatic interactions in its scoring function. Also, water molecules have been removed from the receptor to reduce any potential noise [6].

The conversion process involves using the graphical interface of MGLTools to load the PDB file of the ligand, set up the necessary parameters like charges and atom types, and then

export the file in PDBQT format. This GUI approach makes it accessible for users who may not be familiar with command-line tools, providing a visual method to ensure that all necessary molecular information for docking is correctly formatted and included.

The accuracy of the docking predictions heavily depends on the quality and format of the input data. By ensuring that ligands are correctly prepared and converted into the PDBQT format using the MGLTools GUI, this step is crucial in the computational phase of drug discovery, where precise simulations can lead to more focused and cost-effective laboratory experiments downstream.

Step 4: Defining the Grid Box, Generating Parameter Files, and Configuring Settings

In the fourth step of the docking process, we focus on defining the grid box and generating the necessary parameter files for docking simulations with AutoDock, as well as preparing a configuration file for use with AutoDock Vina. This comprehensive setup is crucial for accurately modeling the docking interactions between the ligand and the receptor.

Grid Box Definition:

The grid box specifies the region within the receptor where the ligands will dock. Encapsulating the grid box as close to the receptor's pocket site as possible ensures all potential binding sites are considered during docking. This also reduces the risk of a ligand docking in open empty space. The parameters of the grid box, including the number of points and spacing in each dimension (x, y, and z), dictate the resolution of the docking simulation and the extent of the search area.

Parameter Files:

- *.GPF (Grid Parameter File)*: Contains settings for the docking grid, such as dimensions, spacing, and scoring functions, configuring the environment for docking.

- *.DPF (Docking Parameter File)*: Details the docking parameters, including the ligand's properties and algorithmic settings, guiding how the ligand interacts during the simulation.

- *Config.txt*: This file is specifically prepared for AutoDock Vina, another popular docking tool that requires a configuration file to run. The config.txt file includes essential parameters such as the center and size of the grid box, the number of modes to be calculated, and energy thresholds. This file ensures that AutoDock Vina operates under optimal settings for the specific docking study, handling input parameters in a way that maximizes computational efficiency and accuracy.

Creating the Parameter and Configuration Files: These files are generated using graphical tools within the MGLTools suite or directly via command-line interfaces, providing flexibility in how researchers set up their docking simulations. The generated files (.gpf, .dpf, and config.txt) are crucial for guiding the docking software in simulating molecular interactions accurately.

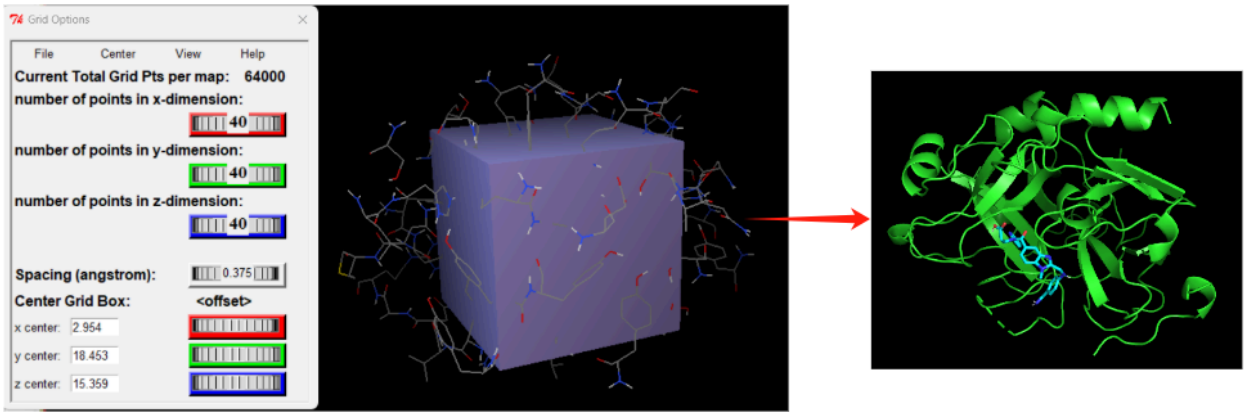

Figure 7: Define grid box

Figure 7 illustrates the configuration of the grid box in a molecular docking setup. The left part of the image displays the graphical interface used for setting up the grid box parameters around the ligand, ensuring all potential interaction sites are included. The right part of the image shows the protein structure in green, with the grid box precisely aligned to encompass the active sites, crucial for effective docking. This visual is instrumental in confirming that the grid box is properly positioned to capture all relevant molecular interactions during the simulation.

This detailed setup, including the creation of both parameter files and a configuration file, is vital for ensuring that the docking simulations focus accurately on the intended areas of the receptor. This careful preparation of the docking environment facilitates accurate and reliable predictions of ligand-receptor binding, enhancing the potential for discovery in drug development projects [6], [32].

Step 5: Running the Docking Simulation and Demonstration

The fifth and final step in the molecular docking process involves executing the docking simulations using AutoDock4 and AutoDock Vina. This crucial phase determines how ligands interact with the target receptor, with the simulations providing initial poses that are crucial for

training graph convolutional neural networks, specifically designed to predict binding affinity.

For AutoDock4, the simulation process is initiated with the AutoGrid4 to calculate the grid parameters using the .gpf file, outputting the results to a log file (.glg). This step prepares the docking environment by setting up an energy grid that the docking algorithm will use. Then the AutoDock4 to perform the actual docking simulation, processing the ligand as specified in the .dpf file and logging the results in a .dlg file. After autogrid and autodock, we extract the docked pose from the .dlg file, convert it to a more general PDB format, and then combine it with the target protein file to form a complete docked complex file. This file visually represents the ligand in its bound state with the receptor [6], [32].

For AutoDock Vina, which is known for its speed and efficiency, the simulation is conducted with specifies the receptor and ligand files, uses a configuration file to set the parameters (such as the search space and docking exhaustiveness), and outputs both a log file and the resulting docked complex in PDB format.

During both types of simulations, monitor the progress to ensure that computations are proceeding as expected without errors. The output file and resulting complex structure are then analyzed to assess the docking outcomes. These results provide the initial poses required for training the neural networks, understanding the ligand's binding efficiency, orientation, and conformation at the binding site.

For a comprehensive understanding of the entire docking process and result interpretation, we provide two demonstration videos:

- AutoDock4 and Vina Steps: https://youtu.be/X6Bu-fNb2_8

- Result Interpretation: https://www.youtube.com/watch?v=i8oYRwvkFMw

These videos offer practical insights into how to set up and run the simulations, as well as how to interpret the results effectively, enhancing both the theoretical knowledge and practical skills required for successful molecular docking. This comprehensive approach not only facilitates the generation of accurate initial poses for neural network training but also advances our understanding and methodology in drug discovery through the application of graph convolutional neural networks in predicting binding affinities.

Dataset visualization

After preparing and curating the dataset for molecular docking simulations, it is crucial to visualize the properties of the dataset to ensure its suitability for further analyses and model training. The focus of the visualizations is exclusively on the ligands, rather than the proteins, for several key reasons. The primary reason for focusing on ligand visualizations in the dataset and not the proteins is due to the nature of the docking simulations where the receptor (protein) is typically considered a fixed entity, while the ligands are variable. In molecular docking studies, especially when preparing data for training models that predict binding affinity, it is essential to understand the diversity and properties of the ligands because they are manipulated during simulations. Ligands must be able to adapt their conformations through transformations such as twists and rotations around bonds to fit into the binding site of the receptor effectively. This adaptability can significantly influence the outcome of the docking process and the subsequent predictive modeling.

In contrast, the protein or receptor structure remains constant in these simulations. The ligands must be assessed for their potential to assume various conformations that align with the

static structure of the protein binding site. Therefore, the variability and complexity of the ligands, such as their rotatable bonds and atomic makeup, are of particular interest, as these factors determine how the ligands can be optimized to fit into the receptor site and form stable complexes.

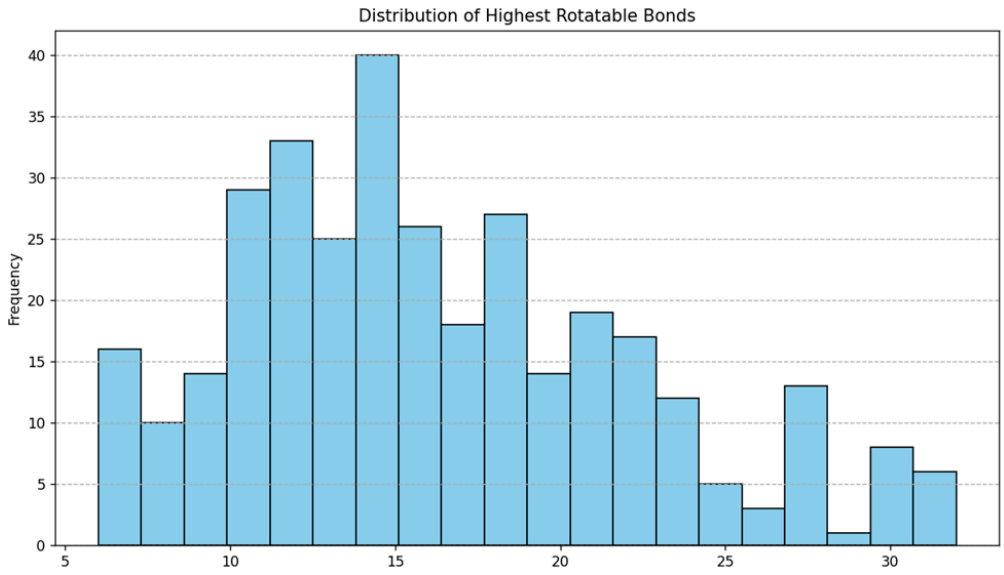

Figure 8: Distribution of Highest Rotatable Bonds

Figure 8 shows the distribution of the highest number of rotatable bonds among the ligands within the prepared dataset. The x-axis, ranging from 5 to 32, reflects the criteria used during the dataset preparation where ligands with fewer than 5 and more than 32 rotatable bonds were excluded. This filtering was applied to focus on ligands with a manageable level of flexibility, which are more suitable for the computational requirements and accuracy needs of the docking software used.

The frequency of ligands within each range of rotatable bonds is plotted on the y-axis. The data shows a peak frequency around 15 - 20 rotatable bonds, indicating that a significant number of ligands within this dataset possess a moderate level of molecular flexibility, which is optimal for effective docking simulations. This peak suggests that most ligands in this set are

neither too rigid nor excessively flexible, making them ideal candidates for achieving reliable docking interactions without overwhelming the computational resources. This visualization helps in understanding the flexibility profile of the ligands, which is crucial for predicting how they might interact with the fixed receptor structure during the docking simulations.

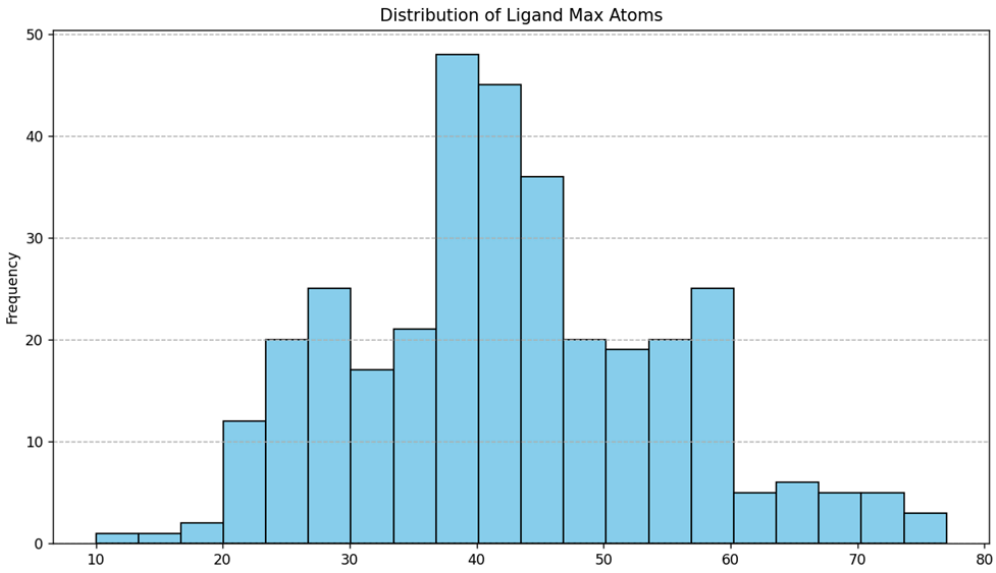

Figure 9: Distribution of Ligand Max Atoms

Figure 9 shows the distribution of the maximum number of atoms per ligand across the dataset. The x-axis represents the total number of atoms in a ligand, highlighting a range that predominantly lies between 20 and 70 atoms. The most frequent category is observed between 30 and 50 atoms, where the histogram peaks, indicating that the majority of the ligands in this dataset feature a moderate level of structural complexity.

This visualization is significant as it informs about the overall size and complexity of the ligands, which directly impacts the computational demands of the docking simulations and the interaction possibilities with the receptor. Ligands with a moderate number of atoms are often preferred in computational studies because they provide sufficient complexity to allow for realistic interactions while not being overly demanding in terms of computational resources. This

balance is crucial for efficiently processing a large set of simulations without compromising the quality of the interactions being modeled.

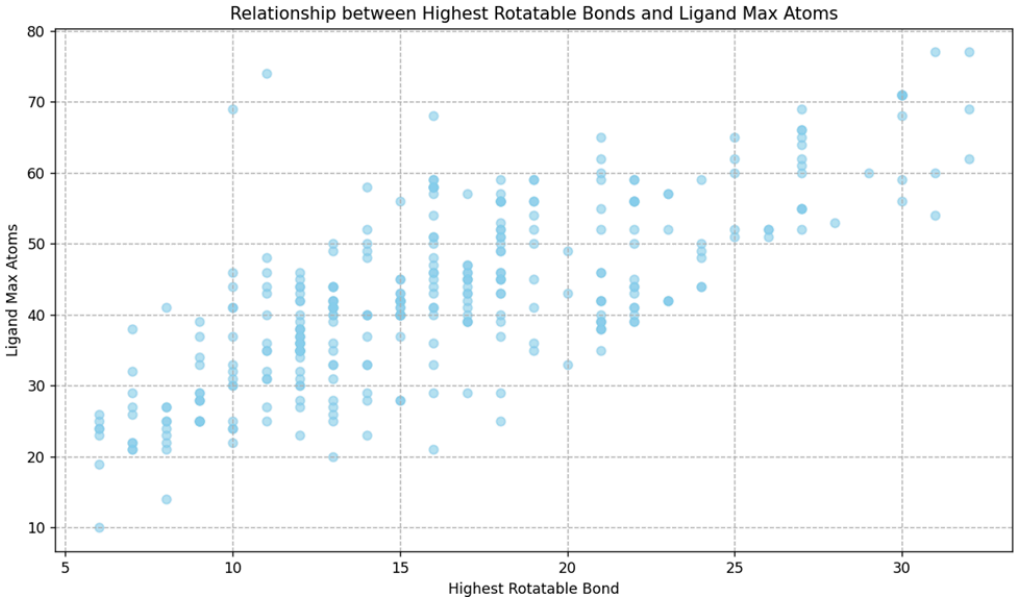

Figure 10: Highest Rotatable Bonds vs Ligand Max Atoms

Figure 10 shows the dependencies of two fundamental ligand features, the number of rotatable bonds, and the number of atoms at the maximum. A ligand's value is represented at all points of the plot in order of the number of rotatable bonds (x-axis) and the total number of atoms of that ligand (y-axis). Here, the data suggests that the ligands with more atoms usually also have a higher number of rotatable bonds, implying that the mass has a direct relationship with the flexibility of the molecule.

This relationship is essential for understanding the dynamic behavior of ligands during docking. Ligands that are both large and flexible (upper right quadrant) may adopt multiple conformations when interacting with a receptor, potentially increasing their chances of finding a favorable binding pose. Conversely, smaller, less flexible ligands (lower left quadrant) might have fewer conformational options, which could limit their binding possibilities but also reduce

the computational complexity of the simulations. By analyzing this plot, we can better predict what kind of ligands might pose computational challenges or require more nuanced handling during the docking process due to their size and flexibility. This understanding is crucial for optimizing docking strategies and ensuring that the computational resources are aligned with the complexity of the ligands being studied.

Parameters to consider for Molecular Docking

a) Molecule Flexibility: Semiflexible approach

Molecular flexibility has to be taken into account to reflect actual binding in a real-world scenario. A semi-flexible approach is taken where the ligand is flexible and the receptor is rigid to mimic a lock and key model as shown in *Figure 11* below. Most studies do this method as it is computationally less intensive. To make the simulation more realistic and incorporate induced-fit theory (ligand changing the receptor's conformational structure as shown in *Figure 12)*, recent developments are considering making both the ligand and receptor flexible. This is typically done by selecting certain residues and side-chains of the receptor to be flexible otherwise it would be too computationally intensive. To know which residues to select, would require extensive sampling of the target's conformational space through molecular dynamics. Given this limitation, the semiflexible approach will be used for simplicity.

A ligand has several degrees of freedom: position (x,y,z), orientation (qx, qy, qz, qw), and torsions aka rotatable bonds ($\tau 1$, $\tau 2$, ... τn). Using these factors can generate a conformational space the ligand can occupy, however, it is computationally infeasible. A sampling method would be needed to optimize exploration and find the best ligand conformation. The stochastic method is the most popular sampling method used for this practice [21].

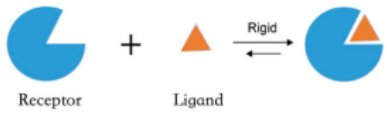

Figure 11: Lock-And-Key Model [5]

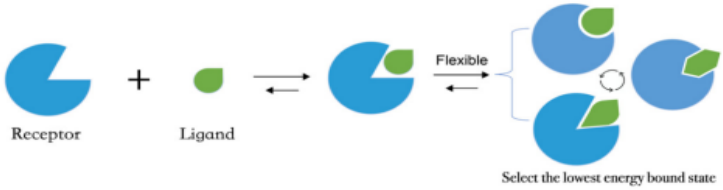

Figure 12: Induced Fit Theory [5]

b) Generate ligand conformations to the receptor

Stochastic method

In this method, a grid box is inputted in a selective portion of an open space where the receptor is located. This optimizes the time and requires less computation. Pseudo-random functions are done to generate ligand conformations to the receptor. To decide what area to explore, the ligand is guided by scoring functions. The higher the score it can find at a particular area in a grid box, that is where it will go. Genetic Algorithm (GA) is the most used pseudo-random function for the stochastic method [21]. A limitation of the stochastic method is that the selection of potential hits in compounds is a random process. As of 2022, there is no consensus on what a compound's binding energy and ligand efficiency (LE) should be. Therefore the search must be repeated and redocked to improve chances of success [22].

Pseudo-random function: Genetic Algorithm

A genetic algorithm (GA) is a global search algorithm that can perform genetic operations. In this case, a random population for a ligand can be set at a range between 50-300. Each of these ligand variants is calculated to see how well it interacts with the receptor. The

best-fit ligands with the best scores are chosen and then reproduced to create slight new variants of that ligand. In a sense, it simulates the process of natural evolution as they mix and mutate genes (a state variable) using crossovers. This process is repeated for a predefined number of generations, with 10 being the default to balance ease of analysis with computational efficiency. With each iteration, the new variants improve their fit within the receptor binding site. Termination occurs once a good fit has been found or if all set numbers of generations have already occurred. A limitation to this is that this would produce accurate results for ligands that have less than 10 rotatable bonds. If it is greater than 10, then either the ligand is broken into 2 pieces to dock separately or another option is to freeze some rotatable bonds [6].

Pseudo-random function: Lamarckian Genetic Algorithm (LGA)

Lamarckian genetic algorithm (LGA) is slightly different from GA where it passes a ligand's genetic info and acquired adaptations gained during its lifetime. This also reproduces slight new ligand variants to fit better to the receptor. Incorporating acquired adaptations allows an introduction of "learning" into an evolutionary process so that adaptation occurs quicker and more efficiently in each generation to find the optimal bind of the receptor. Similar to GA, it produces accurate results for ligands that have at most 10 rotatable bonds. Within LGA to simulate acquired adaptation, there is a Local Search (LS) feature to modify the ligand's phenotype and a Solis and Wets local search to inverse map the phenotype to a genotype [6].

c) Predicting Ligand's Binding Energy Score

Force Fields

Numerous force fields are available such as *AMBER, GROMOS, OPLS,* and *CHARMM,* and are used for accuracy-related atomic distances and separate computing of bound and

unbound complex energies [21]. Since over 90% of structures are solved with X-ray crystallography and are the majority of structures in the Protein Data Bank (PDB), it would be safe to use force fields to energy-minimize the ligands before molecular docking. If the structure originally came from Nuclear magnetic resonance (NMR), then it's not necessary to do this step as NMR already outputs a high-resolution image. For structures that originally came from X-ray crystallography, however, this is a crucial step as applying force fields to the structure would relax the atoms that have been stretched by bonds when they were co-crystallized with the cofactor. In other words, the ligands are too rigid and they need to be flexible so that they can act more realistically and establish better predictions. [13]. Autodock4 implements AMBER as their force field and OPENBABEL software can be used before Autodock4 and Autodock Vina to energy-minimize ligands for further accuracy such as using Merck Molecular Force Field 94 (MMFF94) for small molecule ligand. Force Fields are used alongside both Physics-based and Empirical-based Scoring functions [24], [25].

Predicting Ligand's Binding Energy Scoring Functions (SF)

Since the early 1990s, a variety of scoring functions (SF) have been developed by researchers formulated by different assumptions or algorithms. SF is a computational method that estimates ligand-receptor binding free energy based on a single ligand-receptor complex structure. The score of the favorable pose represents the compound's binding affinity. Binding affinity is determined by its binding free energy [12]. There are 4 categories of scoring functions: (i) physics-based methods, (ii) knowledge-based statistical potentials, (iii) empirical scoring functions, and (iv) machine-learning scoring functions [21]. In this case, physics-based from Autodock4 and empirical scoring from Vina were used.

d) Reducing Computing Time

To reduce computing time for docking, a high-performance computing (HPC) environment should be used to run the programs smoothly. For our in-silico experimentation, the computer used to run simulations is an ASUS TUF Dash F15 (2022) with 12th Gen Intel® Core™ i7-12650H Processor 2.3 GHz, NVIDIA® GeForce RTX™ 3070 Laptop GPU and runs on a Windows 10 operating system. Another factor to consider in reducing computing time is conformational space. Instead of using an entire space to see where the ligand would bind to the receptor, a grid box should be created that is centered only on the receptor protein. That way, it will increase the probability that the ligand will bind to the receptor rather than being by itself in a lost space. There are 2 types of docking to do this:

e) Focus docking vs. Blind Docking

Focused Docking

To do focused docking, one would require prior knowledge of the localization of the target receptor's ideal interaction site that has the lowest binding energy score. A drawback to this however is that for virtual screening, it cannot be generalized or guaranteed that all ligands will bind to the same receptor site [21]. If the ligand was docked in any location other than the known binding cavity, poses and score results would be meaningless [27].

Blind Docking

An advantage of blind docking is it allows one to explore the entire receptor protein if one does not have prior knowledge of the receptor's ideal interaction site and can evaluate trends as to where the ligand binds. A drawback however is that it may impact docking accuracy [21]. Considering the 2 options, blind docking is more commonly used and gives greater accessibility

in viewing the entire receptor so this option will be chosen. Additionally, Using other programs such as AutoLigand, F-Pocket or Q-site Finder can be supplementary used to detect optimal binding sites and save time rather than doing redocking [21].

Summary of Molecular Docking Parameters

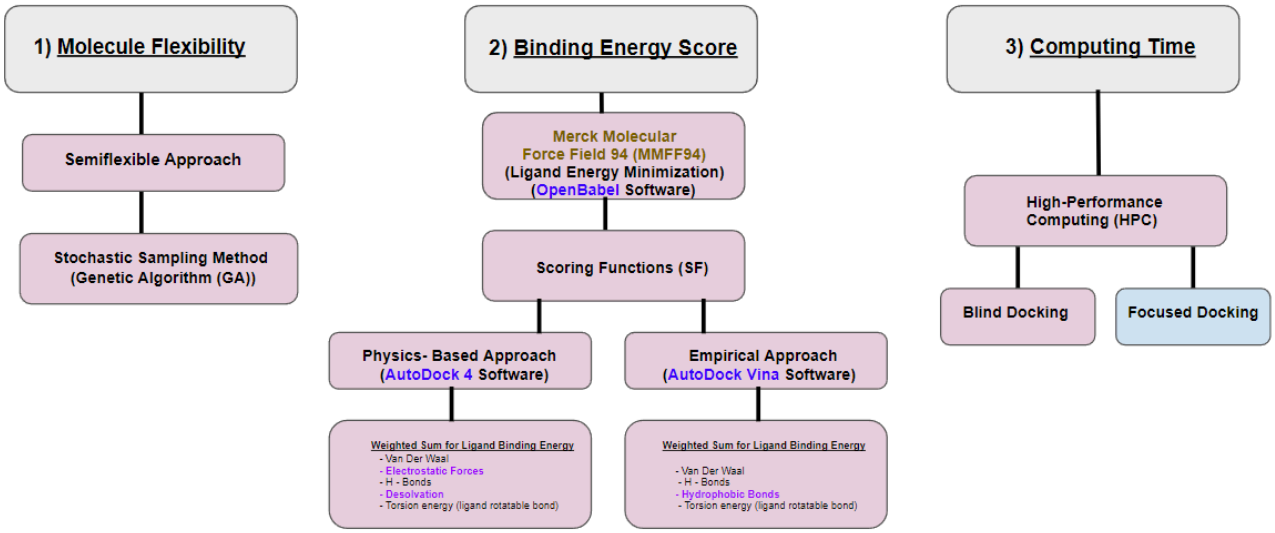

Figure 13: Summary of Parameters Chosen for Molecular Docking

Overall, a summary of 3 parameters are considered for molecular docking as shown in *Figure 13* above, molecule flexibility, binding energy score, and computing time. For molecule flexibility, a semiflexible approach will be taken where the ligand is flexible and the receptor is rigid and a stochastic sampling method will be used with GA and LGA. To get a binding energy score, openbabel software will be used to run an Molecular Force Field 94 (MMFF94) force field for ligand energy minimization. This force field is particularly optimal for small molecule ligands rather than using a more generalized force field like AMBER. From there, both a physics-based and empirical scoring function will be performed using Autodock and Vina software respectively to get the weighted sum of ligand binding energy. All this is done through high-performance computing (HPC) using focused docking as the pocket regions of the receptor

are given in the PDBbind 2017 refined dataset. Focused docking also helps with computational efficiency.

GCN Overview

Artificial Intelligence (AI) / Machine learning (ML) can introduce novel approaches to improve the scoring of a ligand-receptor complex system, by optimizing an existing scoring function or by creating a new scoring function that considers the structure of a complex as input [15]. Graph Convolutional Networks (GCNs) are a type of neural network designed specifically to work with graph-structured data. In molecular docking research, they are excellent at capturing the connections and interactions between several entities, including atoms and molecules. In order to update its representation, each node in a GCN uses a mechanism called message passing, in which it gathers information from its neighbors (other connected atoms or molecules).

To improve the stability of training, especially in deeper networks, GCNs use residual connections. These connections help solve problems like gradient vanishing, which can occur when training deep networks by allowing gradients to flow more easily through the layers.

After several rounds of message passing and aggregating features from neighboring nodes, GCNs often use a technique called global mean pooling. This stage condenses all of the graph's information into a single, compact vector that captures its essence. This vector is then utilized for a variety of purposes, including estimating binding affinities in molecular docking studies and identifying the most promising interactions between proteins and ligands [26].

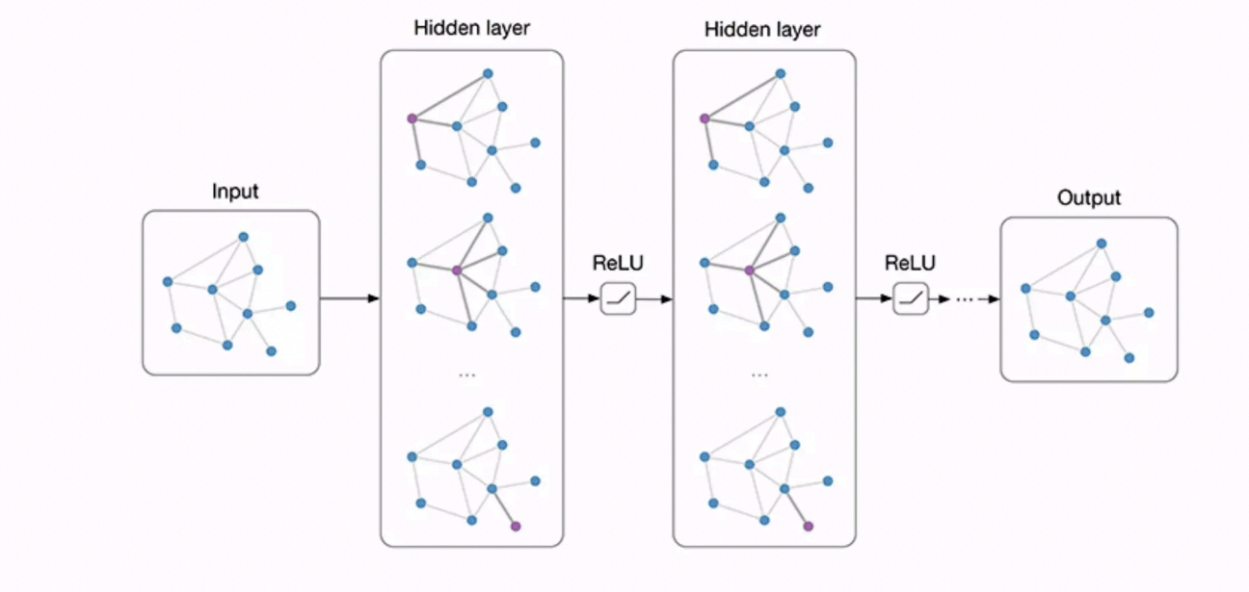

Figure 14: GCN Architecture [29]

Figure 14 explains the structure of a Graph Convolutional Network (GCN) and how it processes molecular data for tasks like predicting binding affinity.

- *Input Layer*: The process starts with a graph representation of the molecule. In this graph, nodes represent atoms, and edges represent interactions such as bonds or spatial proximity between atoms. Each node is initialized with specific features, like the type of atom, its charge, or electronegativity, while edges can include properties like bond type or distance. This graph serves as the starting point, capturing the basic structure of the molecule..

- *Hidden Layers*: The graph is then passed through several graph convolutional layers. In each layer, nodes collect and combine information from their connected neighbors through the edges. This process updates each node's features by combining its own properties with those of neighboring nodes. After the combination, the data undergoes a transformation followed by a ReLU (Rectified Linear Unit) activation, which helps the model learn complex molecular patterns. With each successive layer, the features become more

refined, allowing the network to understand both local interactions and the molecule's overall structure [8].

- *Residual Connections*: To make training more stable and effective, residual connections are added to the network. These connections allow the features from earlier layers to bypass directly into later layers, preventing problems like gradient vanishing and improving the model's performance.

- *Output Layer*: Once the graph passes through all the layers, a global pooling operation (like mean pooling) condenses the information from all nodes into a single vector. This vector represents the entire molecule and encapsulates its structural and interaction details. This final output is then used to make predictions, such as binding affinity or other molecular properties [36].

Unlike traditional methods that rely on predefined rules, GCNs can directly learn molecular patterns from raw graph data. This adaptability makes them highly accurate and versatile for complex tasks, including predicting how molecules interact with proteins.

Graph Construction from Docking Poses

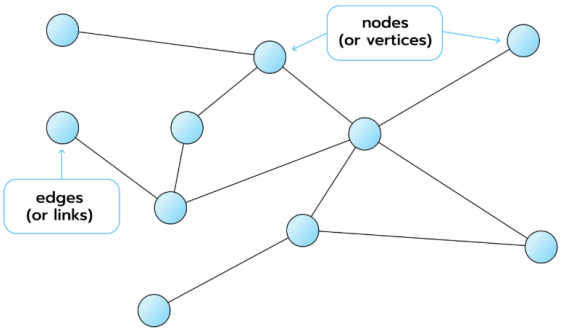

Figure 15: Graph Construction from Docking Poses [34]

Figure 15 shows graph construction from docking poses. To utilize Graph Convolutional Networks (GCNs) in molecular docking, raw docking poses—often generated by tools like AutoDock or Vina—must first be converted into graph representations. This transformation involves several steps that enrich the data with chemical and spatial information, enabling GCNs to analyze molecular interactions effectively. The first is input from docking poses that describe the spatial arrangement of proteins and ligands in their bound states. Each pose represents a potential configuration of how the ligand binds to the protein, complete with interaction details. The nodes represent atoms from both the ligand and receptor target protein. Each node has features that capture unique chemical properties of each atom such as atomic type (e.g., carbon, nitrogen, oxygen), charge and electronegativity values, spatial coordinates in 3D space, and hybridization states or other relevant chemical properties. The edges capture interactions or relationships between atoms. These include bond interactions (e.g., covalent bonds within the same molecule), spatial proximity interactions join atoms if their distance is less than a specific threshold (e.g., 4.5 Å), and non-covalent interactions such as hydrogen bonds, van der Waals forces, and hydrophobic effects. Overall , each edge is enriched with features such as bond type euclidean distance or related metrics like inverse or exponential distance [26].

The resulting graph encodes the entire docking pose and can be stored using adjacency matrices, which outline connections between nodes and feature matrices, which contain the properties of nodes and edges. Each docking pose is converted into its own graph. For a single protein-ligand complex, this process results in multiple graphs—one for each docking pose. The GCN analyzes and ranks these graphs to identify the most favorable binding pose. Each docking pose is converted into its own graph [26].

Experimental Setup

For this project, a High-Performance Computing (HPC) system was utilized to handle the computational demands of training the Graph Convolutional Network (GCN) on molecular docking data. Other setups for the machine learning resources are shown in Table 1 below.

Machine Learning Resources	**Resource Names**
Programming Language	• **Python 3.8**
Deep Learning Frameworks	• **Pytorch**: Implements and trains the GCN model with CUDA for GPU acceleration • **TensorFlow:** Employed for additional performance comparisons and testing other machine learning models.
Tools	• **Biopython**: Utilized for parsing and processing the Protein Data Bank (PDB) files to extract molecular features. • **CUDA**: Leveraged for GPU-accelerated computations during GCN training.
Other Libraries	• **Scikit-Learn**: For data processing and evaluation metrics like precision-recall and precision-recall Area under Curve (PR AUC)

	• **Matplotlib / Seaborn**: Visualizes training metrics, loss curves and performance results. • **NumPy and Pandas**: Handles data transformations and statistical analysis.

Table 1: List of Machine-Learning Resources

Overall Workflow

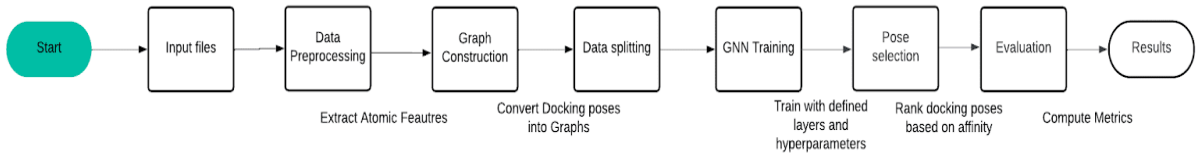

Figure 16: Design workflow

Figure 16 illustrates the design workflow overview. The process begins with input files in PDB (Protein Data Bank) format, which provide detailed atomic structure information for proteins and ligands. A data preprocessing step follows, extracting essential atomic features such as atom type, charge, mass, and electronegativity, as well as covalent bonds between atoms, ensuring clean and consistent data for further processing. The molecular data is then transformed into graph representations, where atoms are nodes and covalent bonds and interactions are edges. This graph data is fed into the GCN, and the dataset is split into training, validation, and test sets. The training set is used to train the model, the validation set fine-tunes the model's parameters, and the test set evaluates the model's generalizability and performance on new unseen data. During the training phase, the GCN is built with specific layers and hyperparameters, such as learning rate and optimizer settings. The model learns the relationships between molecular

features and binding affinities by iteratively updating its parameters to minimize prediction errors and improve accuracy. After training, the model does pose selection, where it ranks docking poses based on predicted binding affinities, selecting the most energetically favorable configurations [26].

An evaluation step follows, where metrics like RMSD (Root Mean Square Deviation) and prediction accuracy are calculated by comparing the model's predictions with experimental ground truth data. Finally, the results are compiled, presenting the predicted binding affinities and evaluation metrics, offering valuable insights into how the GCN-based approach compares to traditional docking methods in terms of accuracy and efficiency [16].

Data Pre-processing

- *Input Format*: Protein and ligand structures are typically supplied in PDB (Protein Data Bank) file format, where they contain extensive atomic-level information, including atom types, spatial coordinates, and energy interactions.

- *Tools Used*: These files are commonly read by tools such as Biopython or Open Babel to extract its molecular data for analysis.

- *Output*: After parsing the PDB files, the extracted data includes a comprehensive list of atoms, properties (such as type and charge), and the connectivity information that defines their bonds or spatial relationships. This structured data is essential for building graph representations and enabling subsequent computational analysis [16].

Node Feature Extraction

In a protein-ligand complex, atoms are nodes in the graph, containing features that

describe its chemical and physical properties.

Key Features for each node includes:

- *Atomic Type*: Atoms are identified by their element type using one-hot encoding. For example, a carbon atom is represented as [1, 0, 0] , while an oxygen atom is [0, 1, 0]. This helps the model differentiate between different elements [33].

- *Atomic Mass*: The atomic mass of each atom is normalized to provide information about its size. For instance, carbon has a mass of 12 u, and oxygen has a mass of 16 u.

- *Charge*: Atoms with partial or formal charges, common in ionic or polar molecules, include this charge information as a feature.

- *Electronegativity*: Each atom's electronegativity, which measures its ability to attract electrons, is scaled and added as a feature. For example, carbon has an electronegativity of 2.55, while oxygen's is 3.44.

- *Spatial Coordinates*: The 3D position of each atom is represented by its (x, y, z) coordinates, capturing the molecule's structure and spatial arrangement.

- *Additional Features*: Depending on the dataset, other information may also be added, such as the hybridization state, valence electrons, or membership in particular functional groups.. These provide further context for the atom's role in the molecule.

By incorporating these features, the graph representation captures a detailed and nuanced view of each atom, enabling the model to better predict molecular behavior and interactions [16], [31].

Edge Feature Extraction

The interactions between atoms or residues in a protein-ligand complex are denoted by edges in a graph representation. A variety of molecular interactions can be captured by the model thanks to these links, which might be either direct bonding or indirect spatial correlations.

- *Bond Type*: If two atoms are directly connected, the type of bond is encoded as an edge feature. For example, bonds can be single, double, triple, or aromatic. This information helps the model understand the nature of chemical connectivity.

- *Distance Threshold*: If an atom's Euclidean distance is less than a given threshold, such as 4.5 Å, it is also regarded as related. This makes it possible for the graph to depict non-covalent interactions that are essential to protein-ligand binding, like hydrogen bonds and van der Waals forces.

- *Distance Metrics*: To quantify the spatial relationship between two atoms, edge features can include:

    - Euclidean Distance: The straight-line distance between two atoms.

    - Inverse Distance ($\frac{1}{d}$): A measure that emphasizes closer interactions.

    - Exponential Distance (e^{-d}): Captures the strength of interaction, with smaller distances contributing more.

- *Interaction Energy*: Additional edge features may include calculated interaction energies, such as electrostatic forces or van der Waals interactions. These provide valuable insights

into the strength and type of interactions between atoms.

By incorporating these edge features, the graph effectively represents both chemical bonds and spatial interactions, enabling the model to learn the complex relationships that govern molecular behavior [31].

Constructing the Graph

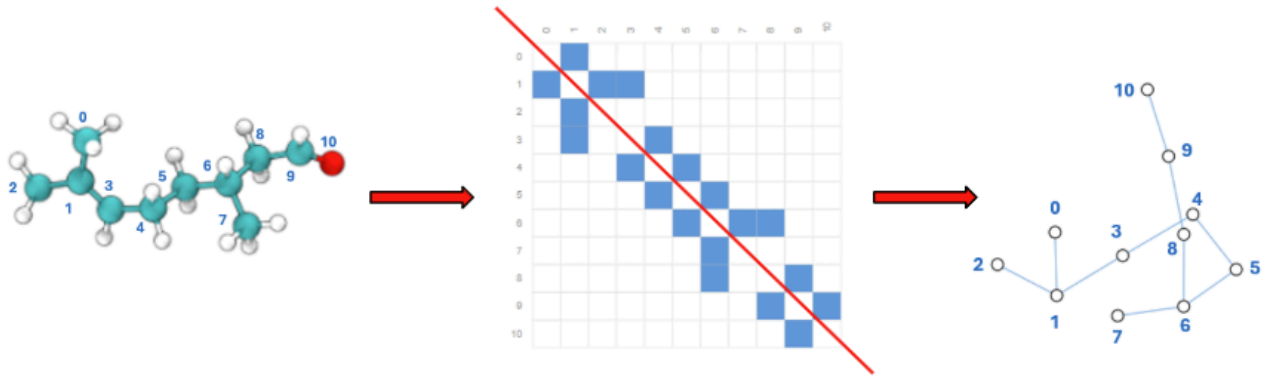

Figure 17: Graph Representation of a molecule [9]

From figure 17 above, the graph is constructed by combining the extracted node and edge features of both ligand and receptor molecule. Each node represents an atom with its chemical and physical properties, while each edge represents either a covalent bond or a spatial interaction between two atoms. This structure captures the molecular relationships comprehensively. Each node is assigned a number for the graph representation. The graph is then saved in an adjacency matrix or edge list, that the model can handle. The matrix represents the connections between each node and edge creating a mirrored graph display if an imaginary linear line were to go through it [9].

To ensure consistency and support efficient model training, all features such as distances, charges, and atomic masses become normalized scaled between 0 and 1. This prevents any single

feature from dominating over another and helps the model converge more effectively during training [16]. By organizing the molecular data into this normalized graph format, the model can efficiently analyze the relationships and interactions within the protein-ligand complex [9].

Data Splitting

After constructing the graphs, the dataset is divided into three sets: training, validation, and test. The training set is used to train the Graph Convolutional Network (GCN) to learn patterns and relationships in the data. The validation set provides feedback during training to fine-tune the model and minimize overfitting by evaluating its performance on previously unseen data. The test set serves for the final evaluation of the GCN, containing completely unexplored protein-ligand complexes. By carefully splitting the dataset, the model's ability to handle unseen molecular interactions is rigorously assessed, ensuring its reliability and robustness [26].

Model Algorithm

Algorithm 1 Protein-Ligand Binding Affinity Prediction with GNN

Dataset D,
distance threshold d,
epochs E,
batch size B,
learning rate α

foreach $f \in D$ **do**
  $x \leftarrow \text{atomic\_features}(f)$
  $\text{edge\_index} \leftarrow \text{atom\_pairs}(f, d)$
  $y \leftarrow \text{binding\_label}(f)$
  $G \leftarrow (x, \text{edge\_index}, y)$

Split G into $G_{\text{train}}, G_{\text{val}}, G_{\text{test}}$

$M \leftarrow \text{GNN}(f_{\text{GCN}}, f_{\text{FC}})$
$\mathcal{L} \leftarrow$ Binary Cross-Entropy Loss
$\mathcal{O} \leftarrow \text{Adam Optimizer}(\alpha)$

for *epoch* $\leftarrow 1$ *to* E **do**
  foreach $b \in G_{train}$ **do**
    $\hat{y} \leftarrow M(b)$ $L_{\text{train}} \leftarrow \mathcal{L}(\hat{y}, b.y)$ Update M using $\mathcal{O}$
  foreach $b \in G_{val}$ **do**
    $\hat{y} \leftarrow M(b)$ $L_{\text{val}} \leftarrow \mathcal{L}(\hat{y}, b.y)$
  Update α using scheduler **if** *early stopping criteria met* **then**
    Break
foreach $b \in G_{test}$ **do**
  $\hat{y} \leftarrow M(b)$
  $L_{\text{test}} \leftarrow \mathcal{L}(\hat{y}, b.y)$
  Append $\hat{y}$ to P
$A_{\text{test}} \leftarrow \text{accuracy}(P, G_{\text{test}}.y)$

Plot $(L_{\text{train}}, L_{\text{val}}, A_{\text{train}}, A_{\text{val}})$
Plot ROC and Precision-Recall curves
Plot P vs $G_{\text{test}}.y$

return M, (L_{test}, A_{test}), P

Figure 18: GCN Algorithm produced with Latex

Figure 18 shows our GCN algorithm with 6 steps total through the process.

- **Step 1: Initialization**
- Purpose: Set up the parameters and prepare the dataset for processing and training.

  Actions:

  1. Define dataset D, distance threshold d, epochs E, batch size B, and learning rate α.
  2. Initialize GNN model M using fully connected layers.
  3. Define loss function $\mathcal{L}$ as Binary Cross-Entropy Loss.

4. Define optimizer O as Adam Optimizer with learning rate α.

- **Step 2: Data Preprocessing**

  Purpose: Process raw input data to extract graph features.

  Actions:

  1. For each file, f belongs to D: f ∈ D

  • Extract atomic features x; includes node features (ie:atom type, electronegativity):

  x ← atomic features (f)

  • Compute edge index by finding atom pairs within distance threshold d.

  edge_index ← atom_pairs (f,d)

  • Assign binding labels y from dataset: y ← binding_label (f)

  • Create graph data: G = (x, edge_index, y) .

  2. Split the dataset G into G_{train} , G_{val} , G_{test}

- **Step 3: Training Loop**

  Purpose: Train model M to minimize loss on G_{train} while validating on G_{val} .

  Actions:

  • For each epoch: epoch ← 1 to E

  • Go into a training Loop:

  • For each batch b that belongs to G_{train}: **foreach** $b \in G_{train}$

  • Compute predicted binding affinities: $\hat{y}$ ← M(b)

  • Calculate training loss: $L_{min} \leftarrow \mathcal{L}(\hat{y}$, b · y)

  • Update model parameters M using optimizer O .

• Go into a Validation Loop:

• For each batch b that belongs to G_{val} : **foreach** $b \in G_{val}$

• Compute predicted binding affinities: $\widehat{y} \leftarrow$ M(b)

• Calculate validation loss: $L_{val} \leftarrow \mathcal{L}(\widehat{y}, b \cdot y)$

• Update learning rate α. using a scheduler.

• If early stopping criteria are met, break the loop.

- **Step 4: Testing Loop**

  Purpose: Evaluate the trained model M on G_{test}

  Actions:

  1. For each batch b that belongs to G_{test} : **foreach** $b \in G_{test}$

  • Compute predicted binding affinities: $\widehat{y} \leftarrow$ M(b)

  • Calculate testing loss: $L_{test} \leftarrow \mathcal{L}(\widehat{y}, b \cdot y)$

  • Append predicted affinities $\widehat{y}$ to predictions list P.

  2. Compute test accuracy $A_{test} \leftarrow$ accuracy (P, $G_{test} \cdot y$)

- **Step 5: Visualization and Results**

  Purpose: Generate plots to analyze training, validation, and testing performance.

  Actions:

  1. Plot training loss L_{test}, validation loss L_{val}, training accuracy A_{train}, and training validation A_{val}

  2. Plot Receiver Operating Characteristic (ROC) and Precision-Recall curves.

3. Plot predicted vs. actual binding affinities: P vs. G_{test} · y

- **Step 6: Return Outputs**

  Purpose: Provide trained model and results.

  Actions: Return M (trained GNN model),

  testing loss and accuracy (L_{test}, A_{test}), and predictions P.

Overall, the following are the key features of the algorithm:

- *Graph Representation*: Encodes protein-ligand interactions as graphs for GNN processing.
- *Message Passing*: Inside GCN layers to aggregate node and edge features for learning.
- *Early Stopping*: Ensures efficient training by halting when no improvement is observed.
- *Evaluation Metrics*: Accuracy, ROC, Precision-Recall, and visualization of predictions [26].

Detailed Architectural Model

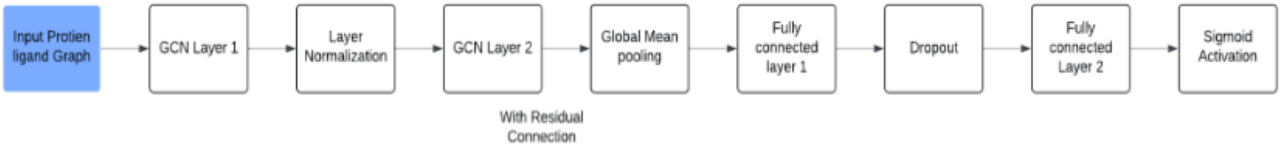

Figure 19: GCN Model Architectural

Figure 19 shows the flow diagram, illustrating only two GCN layers for simplicity. However, in the actual implementation, the model utilizes 8 GCN layers, enabling it to capture higher-order interactions and learn more complex molecular patterns from the graph.

1. Input Protein-Ligand Graph:

The process starts by constructing a graph representation of the protein-ligand complex, where nodes correspond to individual atoms, enriched with features such as atom type, charge, and spatial coordinates. Edges capture atomic interactions, including chemical bonds and proximity-based relationships (e.g., atoms within 4.5 Å). This graph, derived from molecular data obtained through docking simulations or PDB files, encodes both the chemical and structural properties of the complex

2. First GCN Layer: (A message-passing mechanism):

Each node aggregates features from its neighboring nodes via the edges, combining them with its original features. This process allows the model to capture the local chemical environment of each atom, updating node features to reflect both their intrinsic properties and the influence of their immediate neighbors.

3. Layer Normalization:

This process stabilizes feature distributions across all nodes, preventing issues such as exploding or vanishing gradients during training. It also promotes faster convergence by balancing the influence of all features.

4. Second GCN Layer: (Residual Connection):

The updated features from the first GCN layer are passed to the second, which captures more complex molecular patterns by expanding the neighborhood of interactions (e.g., second-order relationships). A residual connection is then introduced, allowing the original input features from the first layer to bypass the second and combine with its output. Residual connections improve gradient flow, retain early layer information, and stabilize training for deeper

networks.

5. *Global Mean Pooling*:

After the GCN layers process the graph, global mean pooling condenses the node-level features into a single vector representing the entire protein-ligand complex. This pooled vector captures both the structure and interactions of the graph, ensuring that predictions are based on the overall complex, rather than isolated atoms or bonds.

6. *Fully Connected Layer 1*:

The graph-level vector is passed to a fully connected (dense) layer, which reduces the dimensionality of the feature vector through a linear transformation. This layer helps the model to identify non-linear patterns in the data and refine its understanding of molecular interactions.

7. *Dropout Layer to prevent overfitting*:

During training, this layer randomly "drops out" a fraction of neurons, forcing the model to rely on diverse subsets of features. This ensures better generalization to unseen protein-ligand complexes.

8. *Fully Connected Layer 2*:

The output from the dropout layer is passed to a second dense layer, which serves as the final transformation step, mapping the feature vector to the desired output dimension. For example, it can output a single scalar value for binding affinity or probabilities for classification tasks.

9. *Sigmoid Activation Function*:

This squashes the output to values between 0 and 1, making it interpretable. For binding affinity prediction, the value represents a normalized score, while for classification tasks, it indicates the likelihood of specific outcomes (e.g., active/inactive binding sites) [26].

Hyperparameter Tuning:

A crucial step in maximizing the GCN model's performance is hyperparameter tuning. It entails choosing the ideal set of hyperparameters to strike a compromise between model generalization and training stability.

Convolutional Layers

The GCN model was set to 8 Convolutional layers.

Learning Rate

Learning rate determines the speed at which the model updates its weights during training. To find the optimal rate that ensured continuous improvement without overshooting the ideal solution, a range of values (e.g., 0.001, 0.005, 0.01) was tested. The most effective learning rate was 0.0009, allowing a steady decline in loss without instability throughout the training epochs.

Batch Size

The batch size determines the number of samples processed before the model's weights are updated. Larger batch sizes (e.g., 128) provided smoother training at the cost of higher memory usage, while smaller batch sizes (e.g., 16 or 32) allowed more frequent updates but introduced less consistency.

Dropout Rate

During training, some neurons are randomly deactivated using the dropout technique to prevent overfitting. Dropout rates ranging from 0.2 to 0.5 were tested, with 0.3 proving to be the

most effective. This rate successfully minimized overfitting without compromising the model's predictive performance.

Optimizer and Learning Rate Scheduler

Hyperparameter tuning balanced the trade-offs between training stability and generalization to unseen data. Higher learning rates led to unstable loss curves, while lower rates slowed training. Adjusting the dropout rate minimized the gap between training and validation loss, enhancing generalization. Selecting an appropriate batch size smoothed training curves and reduced fluctuations in gradient updates [26].

Results

Predicted vs. Actual Binding Affinity

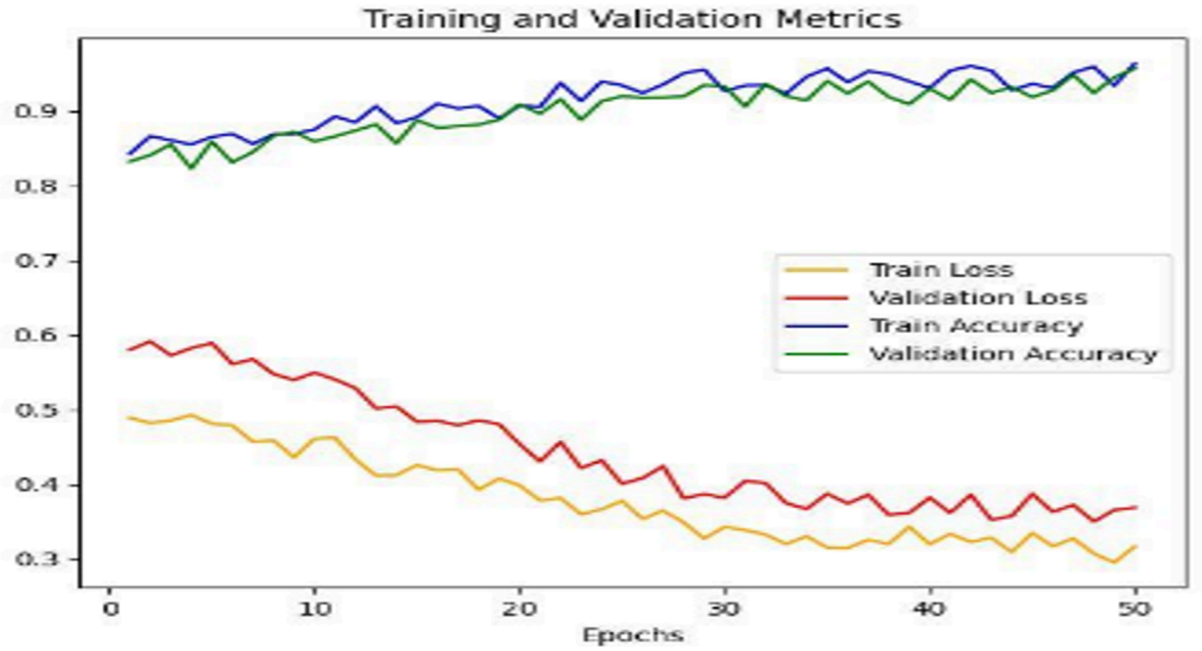

Figure 20: Training and validation results

Since this project worked with 410 files from AD4 and Vina respectivly as per our dataset, 328 files (80% of 410) is the training set and 82 files(20% of 400) is the testing set. From this 80%, *Figure 20* shows the training accuracy (blue) and validation accuracy (green) steadily improve, ultimately reaching 93.97% accuracy after 50 epochs. Fifty epochs indicate that the model has completed 50 full passes through the dataset. The training loss (yellow) and validation loss (red) demonstrate good model convergence, steadily decreasing from approximately 0.6 to 0.3612, with minor fluctuations and spikes. Around 30 epochs, the curves stabilize into a jagged horizontal pattern, indicating that the model has effectively learned patterns from the training data without overfitting. The small gap between the training and validation loss lines suggests good generalization performance. The jaggedness of the lines, more pronounced in the loss curves, may indicate minor training instability, possibly due to the limited size of the dataset. Increasing the dataset size could potentially smooth these curves.

Precision-Recall Curve

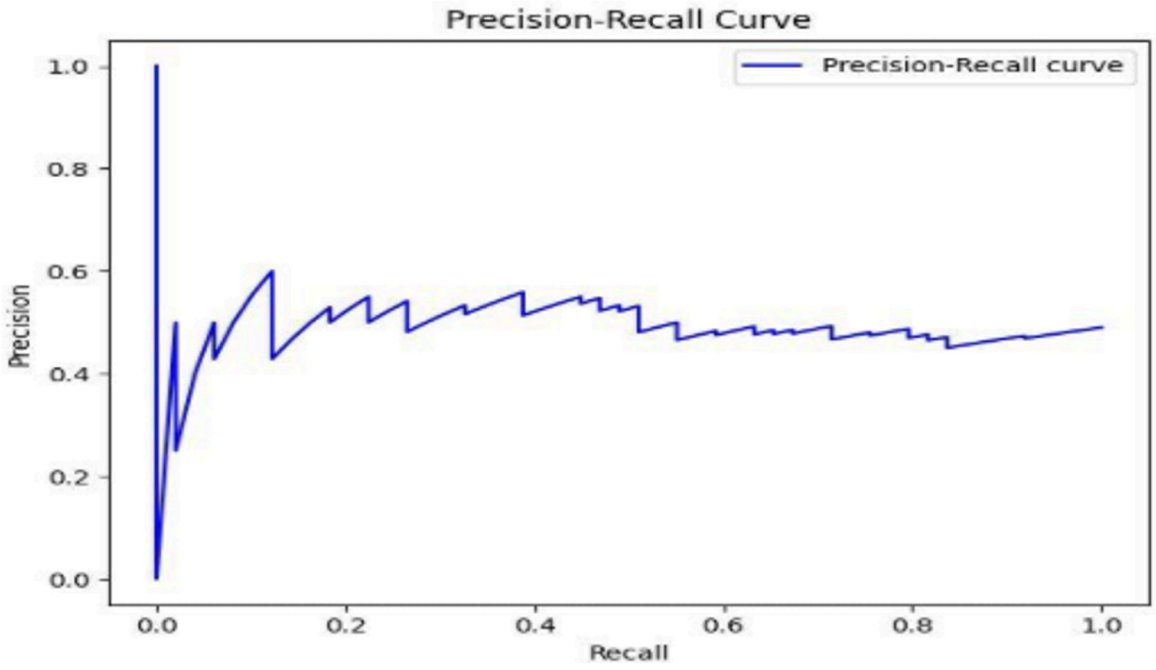

Figure 21: Precision recall curve

Figure 21 shows the Precision-Recall (PR) curve is a crucial metric for evaluating the performance of a classification model. It illustrates the trade-off between precision on the x-axis (the accuracy of positive binding predictions) and recall on the y-axis (the proportion of actual positive binding interactions correctly identified by the model). The curve initially rises sharply, reaching approximately 0.6 precision at around 1 recall, and then stabilizes at about 0.5 precision starting from 0.2 recall. This result is considered moderate, as an ideal PR curve would maintain high precision close to 1 throughout. The area under the Precision-Recall curve (PR AUC) is around 0.5, indicating the model performs better than random guessing but still has significant room for improvement.

Predicted vs Actual Binding Affinity

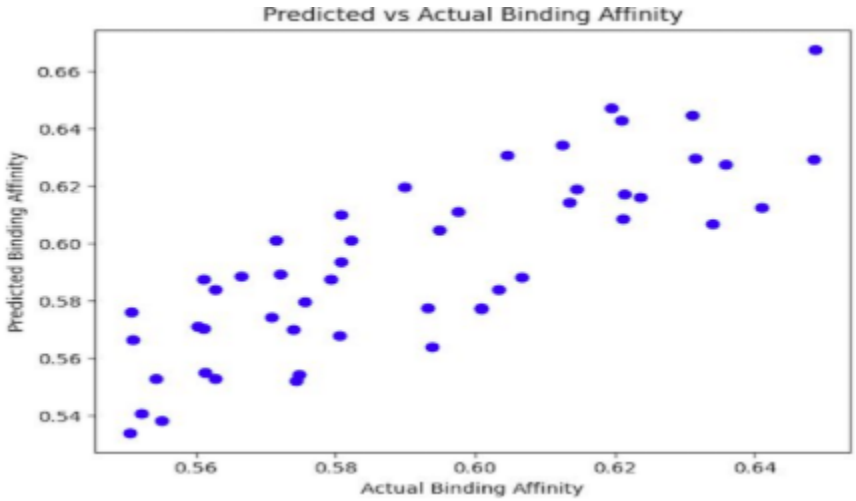

Figure 22: Predicted vs Actual Binding affinity

Figure 22 shows a comparison of predicted binding affinity (y-axis) from the GCN model with actual binding affinity (x-axis) derived from traditional docking tools, AutoDock4 and Vina [18]. The original binding affinity values from AutoDock4 and Vina typically range from -9 to -12 kcal/mol. These values were normalized and transformed into a range between 0 and 1 to facilitate easier comparison, analysis, and grouping within the machine learning model, thereby improving the loss function. Normalization ensures consistency and compatibility with the model's training process. The formula for normalization is as follows:

$$X^{'} = \frac{(X - X_{min})}{(X_{mxx} - X_{min})} \qquad \text{(Equation 7)}$$

Where

$X^{'}$ = Normalized value

X = Original binding affinity value

X_{min} = Minimum binding affinity in dataset

X_{mxn} = Maximum binding affinity in dataset

Most data points are clustered within the 0.54 to 0.66 range. *Figure 22* above shows a positive correlation, with the data forming an upward trend, albeit with some scatter or variability in predictions. This upward trend indicates that the model has successfully learned certain patterns in binding affinity prediction.

Metrics	Value
Training Loss	0.3169
Validation Loss	0.3690
Test Loss	0.3612
Training Accuracy	96.35%
Validation Accuracy	95.67%
Test Accuracy	93.97%
Precision (PR Curve)	~ 0.70
Recall (PR Curve)	~ 0.65

Table 2: GCN - Framework Model Metric results

Table 2 above shows the precise metrics for the loss, accuracy and precision and recall percentages based off from *figures 20* and *21* respectively.

Initial Poses Vs. Traditional Docking Approaches vs. GCN Prediction Model

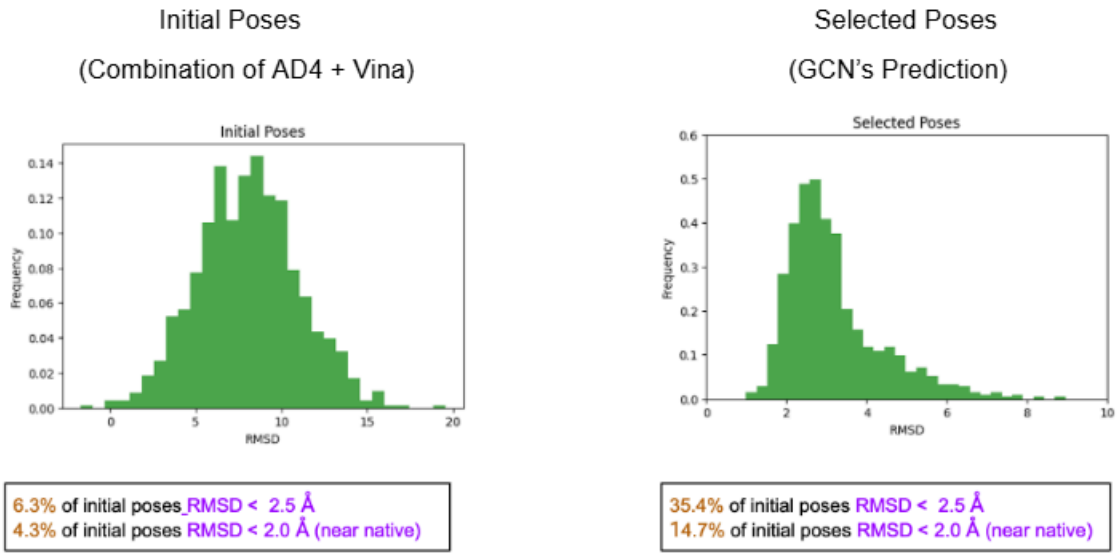

Figure 23: Comparison of results of Initial Poses vs Selected Poses

Docking power reflects the scoring functions (SF) ability to identify native ligand binding poses among computer-generated decoys'. Assessment is done using root-mean-square-deviation (RMSD) to quantitatively measure the similarity between 2 superimposed atomic coordinates of the top-ranked ligand by the SF method and the native ligand pose. A benchmark threshold of RMSD value ≤ 2.5 Å is considered a successful docking as lower RMSD typically indicates a pose closer to the native binding mode, correlating with higher binding affinity [21]. The formula for RMSD is:

$$\mathbf{RMSD} = \sqrt{\frac{\sum_{i=0}^{n} (X_E^i - X_S^i)^2}{n}} \qquad \text{(Equation 8)}$$

Where X_E = Experimental ligand structure's atom coordinates

X_S = Simulated or docked ligand structure's atom coordinates

n = Number of atoms in ligand [21].

Figure 23 compares the initial poses with the selected poses based on average RMSD results amongst a relative frequency distribution. The initial poses are derived from a combination of 410 complexes generated using AutoDock4 and AutoDock Vina, with their RMSD values represented in the left figure. The right figure shows poses predicted by the GCN model.

For the initial poses, the RMSD values range from 0 to 20 Å, with a peak frequency around 7–9 Å. An RMSD < 2.5 Å is considered a good pose, and RMSD < 2.0 Å indicates a near-native pose, signifying low deviation from the ligand's initial crystal pose. Of the 400 complexes, only 6.3% of initial poses have an RMSD < 2.5 Å, and 4.3% achieve RMSD < 2.0 Å.

In contrast, the selected poses predicted by the GCN model (right figure) show an RMSD range of 0 to 10 Å, with a peak frequency at 2–3 Å. The distribution is more concentrated, with 35.4% of poses achieving RMSD < 2.5 Å and 14.7% achieving RMSD < 2.0 Å. This demonstrates that the GCN model significantly improves pose accuracy, achieving a ~5-fold increase in selected poses compared to the initial dataset.

Discussion

Autodock4 and Vina: Software Strengths and limitations

Table 3 below highlights the comparisons between both Autodock4 and Vina software.

	AutoDock 4 (AD4)	**AutoDock Vina**
Type of program	• Freely available open source program • User-friendly - good for beginners	• Freely available open source program • User-friendly - good for beginners
Computer Requirement	• More computationally intensive	• Less computationally intensive
Computer Speed	• Depends on ligand and receptor's atom count • Slow speed calculation: Blind docking: 10 - 20 min Focus docking: 6 - 10 min	• Runs much faster than AD4
Files required to run docking	• .pdb → .pdbqt • .gpf → .glg • .dpf → .dlg	• .pdb → .pdbqt • grid.txt
Scoring Function	• Semi-Empirical Scoring Function • Physics-Based Scoring Function Parameters: • Van Der Waals (Dispersion(Gauss) / Repulsion) • Electrostatic Interaction • H-bond (Improved directionality) • Desolvation (Full model; terms for all atom types) • Torsional Energy (# of rotatable bonds)	• Empirical Scoring Function Parameters: • Van Der Waals (Dispersion(Gauss) / Repulsion) • Electrostatic Interaction **(not included)** • H-bond • Hydrophobic bonds • Desolvation **(not included)** • Torsional Energy (# of rotatable bonds)
Force Field	• Empirical Free Energy Force Field	• Empirical Free Energy Force Field
Search Algorithms	• Monte Carlo Simulated Annealing • Genetic Algorithm (GA) • Lamarckian Genetic Algorithm (LGA) (aka Rapid Hybrid Local Search GA)	• Gradient based search algorithm
Binding Affinity	• General lower scoring than Vina	• General higher scoring than AD4
Binding Poses / Binding Mode prediction	• General higher RMSD than Vina	• General lower RMSD than AD4

Torsions in ligand	• Supports up to 32 torsion (higher torsions → more unstable) (Adjusting to 10 is most accurate)	• Supports up to 32 torsion (higher torsions → more unstable) (Adjusting to 10 is most accurate)
Max Atoms for receptor	• 32768 atoms for Cygwin Unix Environment	• 32768 atoms for Command Prompt
Calibrated from Ligand PDB inhibition constant (Ki) from PDB-bind	• 188 Calibrated complexes (Std Error: 2.52 kcal/mol)	• 1300 Calibrated complexes (Std Error: 2.85 kcal/mol)
Inhibition Constants	• Provides values	• Does not provide values (Need manual calculation)

Table 3: Comparisons and limitations of Autodock4 vs. Autodock Vina

From Table 3 above, each of the two traditional docking software of AD4 and Vina has its strengths and limitations. Typically, Vina runs faster than AutoDock4, mainly due to its less intensive scoring function and the requirement for fewer input files (only a .pdbqt file for both the ligand and receptor, along with a grid.txt file for the gridbox coordinates). While Vina is more computationally efficient, it has drawbacks in its scoring function, as it does not include electrostatic interactions or desolvation in its calculations. To compensate, Kollman charges are manually added to the receptor to account for these interactions. Despite having fewer terms in its scoring function, Vina generally produces lower RMSD values, indicating better posing compared to AD4, whereas AD4 provides lower binding energies due to its more extensive scoring function. Our dataset is limited to small molecule ligands with fewer than 32 torsions and receptor atom counts under 32,768, as this is the threshold supported by both programs. Inhibition constants are not provided by Vina, so they must be manually calculated using Equation 3.

From the result interpretation demonstration video at the end of the molecular docking procedure section of this paper, a limitation was identified in both AD4 and Vina software: the AD4_parameters.dat file provided in the MGL Tools package is restricted to a predefined set of atom types. It excludes certain atoms and metal ions, such as Copper (Cu), Nickel (Ni), and

Potassium (K), which limits the range of receptor molecules that can be used if they contain these atoms. Addressing this issue would require modifying the AD4_parameters.dat file to include these atoms, but this necessitates knowing precise parameter values for each atom. These parameters include the van der Waals radius (Rii), van der Waals well depth (epsilon, epsii), solvation parameter (solpar), atomic volume (vol), hydrogen bond distance (Rij_hb), hydrogen bond well depth (epsij_hb), receptor atom property mapping index (rec_index), hydrogen bonding capabilities (hbond), bond-specific mapping indices, and the atom type mapping in grid maps generated by AutoGrid (map_index) [6]. Determining accurate values for each parameter involves extensive research and validation. Additionally, careful attention must be paid to the formatting of the file to avoid potential issues with software functionality. While this challenge is technically solvable, it would require significant time and effort.

GCN - Model Framework: Strengths and Limitations

Working with protein-ligand systems as graphs comes with several challenges. First, handling large protein-ligand complex systems with thousands of nodes and edges required substantial computational resources. When executed on a High-Performance-Computing (HPC) system, the process took approximately 3 hours to complete. Also, storing and processing graph data for large datasets efficiently is a major challenge. Optimizing memory usage without losing important information is essential for working with large-scale molecular systems. Despite the process taking 3 hours to complete, using HPC is a strong advantage for processing large-scale molecular or graph data efficiently especially when implemented with better RAM space and GUI.

Selecting the appropriate features to include in the graph is critical, as the type and

resolution of features—such as atom properties or interaction metrics—directly impact the model's accuracy and efficiency. Achieving a balance between sufficient detail and computational feasibility is essential. Features not currently included, such as hydrophobicity and solvent-accessible surface area, could enhance the model's ability to simulate more realistic molecular interactions. Incorporating these features may lead to a more accurate representation of real-world molecular scenarios. For future experimentation, feature selection should prioritize atom-level properties (atomic number, partial charges) and interaction metrics (bond type, distances) to capture chemical and spatial relationships. Additionally, incorporating global system characteristics (binding site data, molecular dynamics) and environmental factors (solvent accessibility, pH) will enhance the biological relevance and robustness of the model while maintaining computational feasibility.

Other notable factors include the careful selection of distance thresholds for edge creation to balance accuracy and efficiency; a threshold of 4.5 Å was found to achieve this balance for this model. Additionally, docking simulations can generate unrealistic or erroneous poses that, if unfiltered, introduce noise into the graphs. Methods such as energy thresholding, clustering, and physics-based validation were employed to reduce noise, but effective filtering was not fully achieved due to constraints such as limited validation tools and time. Further efforts are needed to explore validation tools and methods for more effective filtering.

Addressing the ROC-Curve

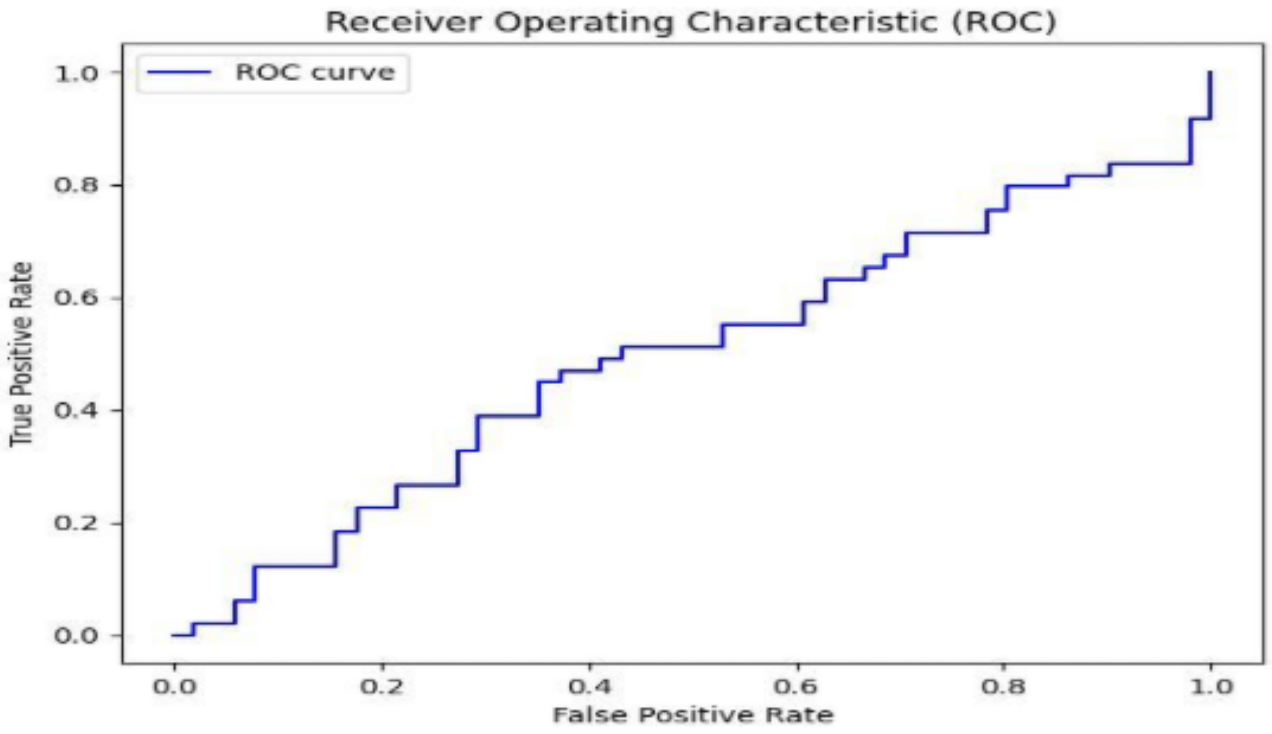

Figure 24: Receiving Operating Characteristic Curve (ROC)

From Table 2, the training accuracy, using Jiang's literature as a benchmark, was 96.35%, while our testing accuracy was 93.97%, indicating a 3% difference in our benchmark. The formula for accuracy is as follows:

$$\text{Accuracy (Acc)} = \frac{t_p + t_n}{total\ samples} = \frac{t_p + t_n}{t_p + t_n + f_n + f_p} \qquad \text{(Equation 9)}$$

where:

- **True Positives (t_p)** - Number of correctly predicted *successful binding interaction.*
- **False positive (f_p)** - Number of incorrectly predicted *successful binding interaction.*
- **True Negative (t_n)** - Number of correctly predicted *no binding interaction.*

- **False Negative (f_n)** - Number of incorrectly predicted *no binding interaction* [11].

Figure 24 presents a receiver operating characteristic (ROC) curve, which evaluates the diagnostic ability of binary classifiers. The x-axis represents the true positive rate (TP) (proportion of correctly predicted successful binding interactions), while the y-axis represents the false positive rate (FP) (proportion of incorrectly predicted successful binding interactions). An ideal curve would have a logarithmic curve approaching true positive at 1 ideally, reflecting high classifier performance [10].

As a baseline, a random classifier produces points along the diagonal line (FP = TP). In Figure 24, the observed curve resembles a stepwise diagonal, suggesting limited discriminative performance. This is consistent with an area under the curve (AUC) value close to 0.5, indicating that the model struggles to effectively distinguish between positive and negative cases. This outcome highlights the potential need for further hyperparameter tuning and the incorporation of additional training data to enhance the model's discriminative capability [30].

It is important to note that while the model achieved a high overall accuracy of 93.97%, this metric alone can be misleading in the context of imbalanced datasets. For instance, when one class dominates the dataset, the model may achieve high accuracy by predominantly predicting the majority class. The ROC curve, therefore, provides valuable insights into the limitations of the model's binary classification performance [30].

However, the ROC curve is not directly applicable to the primary focus of our project, which is on regression tasks predicting binding affinity using graph convolutional networks (GCNs) and assessing their alignment with traditional docking methodologies, as shown in Figure 22 [2]. Here, performance metrics such as mean squared error (MSE) and root mean squared error (RMSE)

provide more relevant indicators of model performance. While the ROC curve highlights areas for improvement in ligand binary classification experiments, our findings remain robust in the context of regression analysis, offering meaningful insights into the predictive potential of GCNs [30].

As a solution to improve the ROC curve, adding a Consensus Assessment of Scoring Functions (CASF) benchmark would be useful to test each specific part of model performance for scoring, ranking, docking and screening to see where the issue may lie in. Another potential solution may be to use CD-HIT software tool and use their sequence identity cutoff set to 0.9 to cluster proteins before doing random splitting. This ensures that the training and testing set do not share similar proteins and accidentally create 1 type of class in the dataset. In this project, random manual selection was done in the PDBbind refined 2017 dataset. Incorporating these tools may lead to an error-less training and testing set selection [16].

Future experimentation:

Comparative analysis of Autodock 4 with GCN and then Vina with GCN alone respectively would be the next steps of experimentation. An 80% train ; 10% validation, 10% testing dataset distribution is the standard version of every ML model implementation. So far we had 410 complexes (328 complexes as 80% training set and 82 as 20% of the testing and validation set) which may be considered a low dataset and not give high enough accuracy if we separated the two traditonal software file inputs rather than combining the two [23]. It would also require further understanding of the code to automatically separate the software files. Due to time constraints, we done comparisons of traditional docking software combining autodock4 and vina to the GCN alone to observe their rmsd and binding affinity outputs. Implementing these next steps with an upper scale preprocessed dataset will help further refine and optimize poses from the GCN model as well as comparing rmsd to Figure 23. Also, according to a study by Ivanova, any ligands that have active rotatable bonds greater than 6 may cause inaccurate results of a high rmsd ("bad" posing) [22]. Future experimentation would be useful in showing how RMSD corresponds to a ligand's rotatable bonds and max atom numbers. Lastly, once a higher dataset is reached and has reputable accuracy with higher percentages of good posing, receptor flexibility in a pocket site region may then be incorporated as a paramter to simulate a real-world protein-ligand binding scenario. However, this would be more computationally exhaustive and would require a strong high performance computer (HPC).

To prepare for graph representation, feature selection should prioritize atom-level properties (atomic number, partial charges) and interaction metrics (bond type, distances) to capture chemical and spatial relationships. Additionally, incorporating global system characteristics (binding site data, molecular dynamics) and environmental factors (solvent accessibility, pH) will enhance the

biological relevance and robustness of the model while maintaining computational feasibility.

Conclusion and Future Directions

A GCN-framework model is proposed to overcome the heuristic scoring functions of traditional docking software such as Autodock4 and Autodock Vina, which may not fully capture the real binding energies and posing. The GCN model can capture rich molecular features by creating graph representation for protein-ligand complexes using atoms, charges and electronegativities as nodes and covalent bonds as edges [16]. This can allow the GCN to learn complex interaction patterns such as hydrogen bonds, hydrophobic interactions, van der waals, desolvation and torsional energies, all of which are dependent on determining binding affinity [6]. The GCN can then rank poses based on predicted binding affinities, helping to identify the most likely binding configurations. Figure 23 has shown a relative frequency distribution of good and bad poses, where the GCN learns from experimental data (410 complexes). This enables the model to predict that 6.3% of the initial poses have an RMSD < 2.5 Å, compared to 35.4% of the GCN-selected poses, resulting in approximately a five-fold increase in good poses for the GCN model. Despite the poses not being able to be fully refined, GCN regardless can prioritize complexes that are energetically favorable. From Figure 22, the model has successfully learned certain patterns in binding affinity prediction from traditional docking methods.

For future experimentation to increase percentages of good posing by refinement, larger and more diverse datasets would need to be trained, allowing for handling of a broader range of molecular scenarios. Additionally, adding more features to the protein-ligand complex may add further accuracy of simulating a realistic protein-ligand binding scenario such as hydrophobicity

and a solvent-accesible surface area as examples to conclude more accurate binding affinities and posing. Once the model has reached a reputable accuracy with higher percentages of good posing, receptor flexibility in a pocket site region may then be incorporated to simulate a real-world protein-ligand binding scenario. These models aim for the goal of optimizing the drug selection process during the discovery and development stages of the drug development life cycle.

Contributions to Our Project

- Dataset Preparation Yolanda Marquez and Yang Liu took the lead in preparing the dataset, essential for training our model. They utilized AutoDock4 and AutoDock Vina to conduct initial docking simulations, which produced a variety of docking poses. Their focused efforts were on optimizing these poses to develop a high-quality dataset, effectively tailoring it to be suitable for input into our GCN model.

- Model Development Supriya Kankati and Akash Jagarlamudi were responsible for the development of the GCN model, specifically crafted for molecular docking tasks. They integrated specialized algorithms to enhance the model's ability to predict binding affinities accurately. Their contributions were pivotal in ensuring the model effectively captured the complex interactions between proteins and ligands.

- As of 2022, most methods have not been proposed and assessed in a complete docking pipeline where classical methods such as Autodock4 (AD4) and Autodock Vina (Vina) are compared to a machine learning workflow. This project aimed to bridge that gap [21].

- Professor Ahmed Hambaba advised the project and helped shape the goal of the project.

References

[1]. A. Ammar, R. Cavill, C. Evelo, and E. Willighagen, “PSnpBind-ML: predicting the effect of binding site mutations on protein-ligand binding affinity,” *J Cheminform*, vol. 15, no. 1, p. 31, Mar. 2023, doi: 10.1186/s13321-023-00701-3.

[2]. A. D. da Silva, G. Bitencourt‐Ferreira, and W. F. de Azevedo, “Taba: A Tool to Analyze the Binding Affinity,” *J Comput Chem*, vol. 41, no. 1, pp. 69–73, Jan. 2020, doi: 10.1002/jcc.26048.

[3]. A. Fersht, *Structure and Mechanism in Protein Science: A Guide to Enzyme Catalysis and Protein Folding*, First Edition. New York: W. H. Freeman, pp. 331, 1998.

[4] A. R. Leach and V. J. Gillet, An introduction to chemoinformatics. Dordrecht: Springer, 2007.

[5]. A. Tripathi and V. A. Bankaitis, “Molecular Docking: From Lock and Key to Combination Lock,” *J Mol Med Clin Appl*, vol. 2, no. 1, p. 10.16966/2575-0305.106, 2017.

[6]. AutoDock4 User Guide: G. M. Morris *et al.*, “AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility,” *J Comput Chem*, vol. 30, no. 16, pp. 2785–2791, Dec. 2009, doi: 10.1002/jcc.21256.

S. Cosconati, S. Forli, A. L. Perryman, R. Harris, D. S. Goodsell, and A. J. Olson, “Virtual Screening with AutoDock: Theory and Practice,” *Expert Opin Drug Discov*, vol. 5, no. 6, pp. 597–607, Jun. 2010, doi:10.1517/17460441.2010.484460.

S. Forli and A. J. Olson, “A forcefield with discrete displaceable waters and desolvation entropy for hydrated ligand docking,” *J Med Chem*, vol. 55, no. 2, pp. 623–638, Jan. 2012, doi: 10.1021/jm2005145.

[7]. Autodock Vina Software:

O. Trott and A. J. Olson, “AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading,” *Journal of computational chemistry*, vol. 31, no. 2, p. 455, Jan.2010, doi: 10.1002/jcc.21334.

Molecular Docking Tutorial: AUTODOCK VINA - PART 1 | Beginners to Advanced, (Mar. 20, 2020). Accessed: Apr. 23, 2024. [Online Video]. Available:https://www.youtube.com/watch?v=Sux91FJ3Xe8

Molecular Docking Tutorial: AUTODOCK VINA - PART 2 | Beginners to Advanced, (Mar. 20, 2020). Accessed: Apr. 23, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=vU2aNuP3Y8I

[8] B. Krishnamurthy, “ReLU Activation Function Explained,” Built In. Accessed: Dec. 15, 2024. [Online]. Available: https://builtin.com/machine-learning/relu-activation-function

[9] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, "A Gentle Introduction to Graph Neural Networks," *Distill*, vol. 6, no. 9, p. e33, Sep. 2021, doi: 10.23915/distill.00033.

[10] C. Chan, "What is a ROC Curve - How to Interpret ROC Curves - Displayr." Accessed: Dec. 14, 2024. [Online]. Available: https://www.displayr.com/what-is-a-roc-curve-how-to-interpret-it/

[11] "Classification: Accuracy, recall, precision, and related metrics | Machine Learning | Google for Developers." Accessed: Dec. 14, 2024. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

[12]. C. Yang, E. A. Chen, and Y. Zhang, "Protein–Ligand Docking in the Machine-Learning Era," *Molecules*, vol. 27, no. 14, p. 4568, Jul. 2022, doi: 10.3390/molecules27144568.

[13]. *Energy Minimization of Proteins and Ligands Pre-Docking Works (#1)*, (Jun. 18, 2020). Accessed: Mar. 13, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=BXuIkIOJKrM

[14] G. M. Morris et al., "AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility," Journal of Computational Chemistry, vol. 30, no. 16, pp. 2785–2791, Dec. 2009, doi: https://doi.org/10.1002/jcc.21256.

[15]. G. S. Heck, V. O. Pintro, R. R. Pereira, M. B. de Ávila, N. M. B. Levin, and W. F. de Azevedo, "Supervised Machine Learning Methods Applied to Predict Ligand- Binding Affinity," *Curr Med Chem*, vol. 24, no. 23, Sep.2017, doi: 10.2174/0929867324666170623092503.

[16] H. Jiang *et al.*, "Predicting Protein–Ligand Docking Structure with Graph Neural Network," *J. Chem. Inf. Model.*, vol. 62, no. 12, pp. 2923–2932, Jun. 2022, doi: 10.1021/acs.jcim.2c00127.

[17] "Journal of Computational Chemistry," Journal of Computational Chemistry, Mar. 2006, doi: https://doi.org/10.1002/(issn)1096-987x.

[18]. J. Wong, "Linear Regression Explained," Medium. Accessed: May 07, 2024. [Online]. Available:https://towardsdatascience.com/linear-regression-explained-1b36f97b7572

[19]. J.M. Stokes et al., "Application of machine learning techniques in drug repurposing for COVID-19," Nature Communications, vol. 11, Article 1638, 2020.

[20]. J. Zhou et al., "Machine learning predictions of antidepressant responses," Journal of Psychiatric Research, vol. 122, pp. 22-28, 2020.

[21]. K. Crampon, A. Giorkallos, M. Deldossi, S. Baud, and L. A. Steffenel, "Machine-learning methods for ligand–protein molecular docking," *Drug Discovery Today*, vol. 27, no. 1, pp. 151–164, Jan. 2022, doi:10.1016/j.drudis.2021.09.007.

[22]. L. Ivanova and M. Karelson, "The Impact of Software Used and the Type of Target Protein on

Molecular Docking Accuracy," *Molecules*, vol. 27, no. 24, p. 9041, Dec. 2022, doi: 10.3390/molecules27249041.

[23]. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, in KDD '16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144. doi:10.1145/2939672.2939778.

[24]. N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison, "Open Babel: An open chemical toolbox," *Journal of Cheminformatics*, vol. 3, no. 1, p. 33, Oct. 2011, doi: 10.1186/1758-2946-3-33.

[25]. "Releases · openbabel/openbabel," GitHub. Accessed: Mar. 13, 2024. [Online]. Available: https://github.com/openbabel/openbabel/releases

[26]. R. Mercado *et al.*, "Graph networks for molecular design," *Mach. Learn.: Sci. Technol.*, vol. 2, no. 2, p. 025023, Mar. 2021, doi: 10.1088/2632-2153/abcf91.

[27]. S. Forli, R. Huey, M. E. Pique, M. F. Sanner, D. S. Goodsell, and A. J. Olson, "Computational protein-ligand docking and virtual drug screening with the AutoDock suite," *Nat Protoc*, vol. 11, no. 5, pp. 905–919, May 2016, doi: 10.1038/nprot.2016.051.

[28]. S. Gu *et al.*, "Can molecular dynamics simulations improve predictions of protein-ligand binding affinity with machine learning?," *Brief Bioinform*, vol. 24, no. 2, Mar. 2023, doi: 10.1093/bib/bbad008.

[29] S. Karagiannakos, "Graph Neural Networks - An overview," AI Summer. Accessed: Dec. 15, 2024. [Online]. Available: https://theaisummer.com/Graph_Neural_Networks/

[30] Ş. K. Çorbacıoğlu and G. Aksel, "Receiver operating characteristic curve analysis in diagnostic accuracy studies: A guide to interpreting the area under the curve value," *Turk J Emerg Med*, vol. 23, no. 4, pp. 195–198, Oct. 2023, doi: 10.4103/tjem.tjem_182_23.

[31] S. Lower, "9.1: Three Views of Chemical Bonding," Chemistry LibreTexts. Accessed: Dec. 15, 2024. [Online]. Available: https://chem.libretexts.org/Bookshelves/General_Chemistry/Chem1_(Lower)/09%3A_Chemical_Bonding_and_Molecular_Structure/9.01%3A_Three_Views_of_Chemical_Bonding

[32] S. Mohd. D. Rizvi, S. Shakil, and Mohd. Haneef, "A simple click by click protocol to perform docking: AutoDock 4.2 made easy for non-bioinformaticians," *EXCLI J*, vol. 12, pp. 831–857, Sep. 2013.

[33] *"T021 · One-Hot Encoding," TeachOpenCADD. Accessed: Dec. 15, 2024. [Online].* Available: *https://projects.volkamerlab.org/teachopencadd/talktorials/T021_one_hot_encoding.html*

[34] “Tapping Into The Power Of Graph Neural Networks – Avenga.” Accessed: Dec. 15, 2024. [Online]. Available: https://www.avenga.com/magazine/graph-neural-networks-and-graph-convolutional-networks/

[35] “The 5 Drug Development Phases,” ThermoFisher Scientific. Accessed: Dec. 12, 2024. [Online]. Available: https://www.patheon.com/us/en/insights-resources/blog/drug-development-phases.html

[36]. T. Tanebe and T. Ishida, “End-to-End Learning Based Compound Activity Prediction Using Binding Pocket Information,” 2019, pp. 226–234. doi: 10.1007/978-3-030-26969-2_21.

[37]. “What does the Ki (inhibition constant) for a drug mean?” Simple and Practical Mental Health. Accessed: Apr. 10, 2024. [Online]. Available: https://simpleandpractical.com/ki-inhibition-constant-receptor-binding-affinity/