POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

A Novel Hybrid Scoring Function for Extreme-Scale Virtual Screening in Drug Discovery

Tesi di Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: **ZHANG YUEDONG**

Student ID: 971513
Advisor: Prof. Gianluca Palermo
Co-advisors: Davide Gadioli, Gianmarco Accordi
Academic Year: 2022-23

Abstract

Drug discovery, traditionally challenged by lengthy development cycles, high costs, and substantial failure rates, has seen alleviation of these issues through the integration of various in-silico methods. This thesis primarily contributes to the Lead Identification stage of In-Silico Drug Discovery by introducing a Hybrid Scoring Function, DrugXGBScore. It is designed to assess Protein-Ligand Complex conformations generated during the Molecular Docking phase of Structure-Based Virtual Screening (SBVS). This function efficiently identifies a set of optimal ligands for a target protein from a large-scale drug molecule database, subsequently facilitating their advancement to wet laboratory experimental testing. Differentiating itself from other scoring functions, DrugXGBScore is specifically engineered to provide not only acceptable prediction accuracy but also an optimal balance with computing performance. This dual objective is attained through a two-pronged approach: algorithmically, by integrating Knowledge-based and Machine-learning scoring functions, and technologically, by seamlessly incorporating DrugXGBScore into our custom-designed High-Performance Computing (HPC) pipeline. The hybrid approach utilizes the straightforward structure of Knowledge-based methods, ensuring smooth integration with our HPC pipeline, while the Machine-learning component is designed to further enhance prediction accuracy. Additionally, the utilization of a high-performance GPU within our HPC pipeline plays a pivotal role in fulfilling our comprehensive HPC objectives. Based on the final evaluation results, DrugXGBScore not only achieved acceptable levels of prediction accuracy but also demonstrated remarkable computational performance. In a case study involving a target protein comprising 8,313 atoms, our HPC pipeline achieved an impressive throughput of 3,347 ligands per second while screening 28,500 decoys. This represents a performance enhancement of approximately four orders of magnitude compared to CPU-only processing.

Keywords: In-Silico Drug Discovery, High-Throughput Virtual Screening, High-Performance Computing, Hybrid Scoring Function

Abstract in lingua italiana

La scoperta di farmaci, tradizionalmente composta da cicli di sviluppo lunghi, costi elevati e tassi di fallimento sostanziali, ha visto un alleggerimento di questi problemi attraverso l'integrazione di vari metodi in silico. Questa tesi contribuisce principalmente alla fase di Lead Identification nell'ambito della scoperta di farmaci in silico, introducendo una Scoring Function Ibrida, DrugXGBScore. È progettata per valutare le conformazioni del complesso Proteina-Ligando generate durante la fase di docking molecolare dello Structure-Based Virtual Screening. Questa funzione identifica efficientemente un insieme ottimale di ligandi per una proteina bersaglio da un'ampia database di molecole farmaceutiche, facilitando successivamente il loro avanzamento ai test sperimentali in laboratorio. Differenziandosi da altre Scoring Functions, DrugXGBScore è specificamente progettata per fornire un tradeoff ottimale tra l'accuratezza della previsione e le prestazioni di calcolo. Questo duplice obiettivo è raggiunto attraverso un approccio bifronte: algoritmicamente, integrando Knowledge-based e Machine Learning Scoring Functions, e tecnologicamente, incorporando senza problemi DrugXGBScore nella nostra pipeline di Calcolo ad Alte Prestazioni (HPC) personalizzata. L'approccio ibrido utilizza la struttura semplice dei metodi Knowledge-based, assicurando un'integrazione fluida con la nostra pipeline HPC, mentre la componente di Machine Learning è progettata per migliorare ulteriormente l'accuratezza della previsione. Inoltre, l'utilizzo di una GPU ad alte prestazioni all'interno della nostra pipeline HPC gioca un ruolo fondamentale nel realizzare i nostri obiettivi complessivi di HPC. Basandosi sui risultati della valutazione finale, DrugXGBScore non solo ha raggiunto livelli accettabili di precisione di previsione ma ha anche dimostrato notevoli prestazioni di calcolo. In uno caso di studio che coinvolge una proteina bersaglio composta da 8.313 atomi, la nostra pipeline HPC ha raggiunto un impressionante throughput di 3.347 ligandi al secondo durante lo screening di 28.500 decoys. Questo rappresenta un miglioramento delle prestazioni di circa quattro ordini di grandezza rispetto all'elaborazione solo CPU.

Parole Chiave: Scoperta di Farmaci In-Silico, Screening Virtuale ad Alto Rendimento, Calcolo ad Alte Prestazioni, Funzione di Valutazione Ibrida

Contents

Introduction

Traditional methods of drug discovery and development, which span stages from target identification to clinical trials, are characterized by high risk, time-intensive processes, and prohibitive costs. Drawing from the detailed research conducted by DiMasi et al. [20] during the early 21st century, it was discerned that the anticipated expenditure for the entire development and subsequent approval of a new drug is approximately US$ 900 million. For the R&D period, the authors noted that if research and development began in 2001, approval would be expected roughly 12 years later. In addition, the drug development process exhibits a notable failure rate. According to a report from the Center of Medical Research (CMR) spanning the years 2008 and 2009, the failure rate of phase II clinical trials alone was as high as 82% [37].
Fortunately, the advent of computational methods in drug discovery and development during the 21st century can mitigate the aforementioned issues. As an illustrative example, during the COVID-19 pandemic, Gadioli et al. [26] introduced the EXSCALATE molecular docking platform for High-Performance Virtual Screening of 15 binding sites of 12 SARS-CoV-2 viral proteins. They deployed EXSCALATE on two modern supercomputers, CINECA-M100 and ENI-HPC5, achieving a combined computational power of 81 PFLOPS. Remarkably, they completed the screening of all 70 billion small molecules in just 60 hours, performing over one trillion ligand-pocket evaluations and setting a new record in the scale of virtual screening.
In the aforementioned example, the concept of High-Throughput Virtual Screening (HTVS) is highlighted. It is a prominent computational technique within the realm of In-Silico Drug Discovery. Utilizing the capabilities of High-Performance Computing (HPC), it swiftly screens extensive compound libraries to identify potential drug candidates to forward to in-vitro experiments. Here, 'virtual' signifies that the screening is executed within a computational milieu rather than a traditional laboratory setting. This computational screening is facilitated by specialized software, such as 'Molecule Docking'. These programs typically begin by sampling the conformation of a complex, which consists of target macromolecules (such as proteins, proteases, or nucleic acids, representing diseases) and small molecule ligands (representing potential drug candidates). They then use a math-

ematical model, known as a Scoring Function, to evaluate the binding affinity, which indicates the strength of the interaction between the molecules. A higher binding affinity signifies a stronger interaction. This thesis primarily centers around the Scoring Function, which is essential for the effectiveness and success of High-Throughput Virtual Screening (HTVS). Inaccuracies in the scoring function can compromise the entire HTVS process, while inefficiencies may hinder evaluating sufficient ligands within the time budget. Therefore, both accuracy and efficiency are of paramount importance for a scoring function.

In literature, Scoring Functions are typically classified into four categories: Physics-Based, Empirical, Knowledge-Based, and Machine Learning Scoring Functions [42]. However, most existing Scoring Functions compete with each other in prediction accuracy, while ignoring the importance of computational performance. Therefore, our work intends to introduce a new scoring function that considers both prediction accuracy and computational performance. Notably, since accuracy and computational performance often stand in opposition at the algorithm level, striking an optimal balance between them is crucial. Simultaneously, maximizing the utilization of available hardware and infrastructure within budgetary constraints is also a key consideration.

This study introduces a novel scoring function designed to balance prediction accuracy with computational performance. It addresses these aspects both at the algorithmic level and through the utilization of advanced hardware capabilities, particularly GPUs. Our primary objective is to achieve satisfactory accuracy while significantly enhancing computational performance, thereby enabling the rapid screening of extensive compound libraries. The scoring function we introduce, named DrugXGBScore, is a hybrid that adeptly merges the advantages of both knowledge-based and machine-learning scoring functions. "DrugXGBScore" is an acronym for Drugscore and XGBoost. More specifically, to achieve our objective, we integrated the optimized Drugscore2018, a Knowledge-based scoring function, with the recently popular machine learning algorithm eXtreme Gradient Boosting (XGBoost). Drugscore2018's simple structure allows for easy and efficient integration into our High-Performance Computing pipeline, while XGBoost contributes to further enhancing the prediction accuracy. The foundational concept of this combined model is partially inspired by the research presented in two articles: one by Wang et al. published in 2017 [68], and another by Lu et al. published in 2019 [43]. They combined two distinct types of scoring functions to enhance the overall prediction accuracy in their studies. To further enhance the computational efficiency, we custom-designed an HPC pipeline specifically for DrugXGBScore, harnessing the power of our advanced Nvidia A-100 high-performance GPU to achieve our High-Performance Computing objectives.

Our scoring function, DrugXGBScore, was evaluated for predictive accuracy using the Comparative Assessment of Scoring Functions 2016 (CASF-2016) benchmark. CASF-

2016 provides a comprehensive framework for assessing the performance of various scoring functions, highlighting their strengths and weaknesses. It employs four key metrics: scoring power, ranking power, docking power, and screening power. These metrics collectively offer insights into the prediction accuracy of a scoring function [63]. For computing performance, we employ various indicators for monitoring, including CPU/GPU utilization, GPU memory usage, and throughput. Within the CUDA segment, the CUDA Profiling Tools are utilized to assess GPU performance. Finally, our comprehensive evaluation demonstrates that DrugXGBScore attains a mid-to-upper tier accuracy compared to other scoring functions in the CASF-2016 Power tests. In terms of computational performance, it remarkably screens all 28,500 decoys of a protein with approximately 8,000 atoms in just 8 seconds. For comparison, performing the same task using only a CPU would require nearly 27 hours, highlighting the significant efficiency of our HPC pipeline.
Let's provide an overview of the structure of this thesis, which spans five chapters. Chapter 1 presents key concepts such as In-Silico Drug Discovery and High-Performance Computing (HPC). Chapter 2 reviews state-of-the-art technologies, highlighting developments like Drugscore, among others. In Chapter 3, we detail our contributions, including our optimization of Drugscore, its integration with XGBoost, and incorporation into the HPC pipeline. Chapter 4 showcases our experimental results, highlighting the prediction accuracy of our algorithm in comparison with other scoring functions. It also emphasizes the enhanced computational performance of our HPC pipeline relative to using only CPUs. The final Chapter 5 concludes our findings and outlines future research directions inspired by this thesis.

1 | Background

In this chapter, we provide the foundational knowledge underpinning this thesis, primarily focusing on two domains: In-Silico Drug Discovery and High-Performance Computing (HPC). For In-Silico Drug Discovery, we adopt a top-down approach that begins with an overview of the Drug Discovery process, progresses through Virtual Screening, and delves into the Molecular Docking phase. We then introduce the tools employed in our research, including Extended-Connectivity Fingerprints (ECFPs), the source database for training and testing our model, and the CASF-2016 benchmark for evaluating the prediction accuracy of our proposed Hybrid Scoring Function. On the High-Performance Computing front, we present a structured level model to provide a logical and architectural exposition of HPC. We hope this level model can enable readers to achieve both a broad and elevated comprehension of the thesis topics.

1.1. In-Silico Drug Discovery

In the modern landscape of pharmaceutical research, "In-Silico Drug Discovery" emerges as a powerful and revolutionary approach. Stemming from the Latin term "in silico", which means "in silicon", it alludes to the use of computer and computational strategies for drug discovery, evoking the idea of performing experiments in the virtual realm rather than in biological test tubes or on petri dishes.
Traditionally, drug discovery was a lengthy, expensive, and often hit-or-miss process, where numerous compounds were synthesized and then tested for their therapeutic efficacy. With the exponential growth in computational power and the development of sophisticated algorithms, the landscape of drug discovery has been transformed. Instead of relying solely on conventional lab-based methods, researchers now harness the power of computers for initial drug candidate screening. Those molecules that show promise in computational tests are then taken forward for subsequent in-vitro evaluations. This integration of computational and experimental methods has led to the birth and rapid evolution of in-silico drug discovery.
The objective of the in-silico drug discovery process is to identify a small set of potential

drug molecules. These molecules could either be novel or existing ones identified through the drug repositioning process [55]. Regardless of its origin, the molecule should bind to a specific target implicated in disease onset [9] and influence its activity in a therapeutically beneficial manner. This process unfolds in a multi-step approach, consisting of four main stages: target identification, target validation, lead identification, and lead optimization. According to Georg C. Terstappen and Angelo Reggiani [65], the drug discovery process is divided into early and late phases. The early phase encompasses target and lead discovery, while the late phase focuses on clinical evaluation and development, as illustrated in Figure 1.1. In-silico methods are predominantly utilized in the early phase of drug discovery.

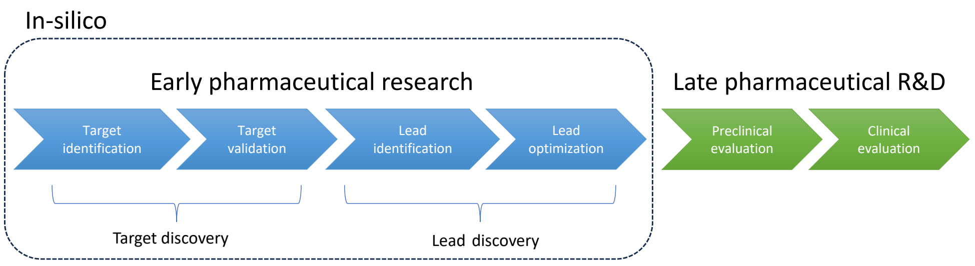


Figure 1.1: Drug discovery process as proposed by Georg C. Terstappen and Angelo Reggiani [65].

The **early pharmaceutical research phase** encompasses target and lead discovery. Targets typically refer to biological macromolecules, including nucleic acids and proteins such as receptors, transporters, enzymes, and ion channels [9]. Leads, on the other hand, are novel chemical molecules identified to act on these targets [65]. The drug discovery process typically begins by selecting suitable drug targets. It's essential to validate these targets, ensuring with a high level of 'confidence' that they have a genuine relation to the disease in question. Once a target is validated, the focus shifts to 'lead identification'. This phase is about finding compounds that can modify the target's behavior. For example, with receptors, these might be activators (agonists) or deactivators (antagonists). For enzymes, they could be boosters (activators) or suppressors (inhibitors), and for ion channels, the compounds might act as openers or blockers [65]. While the concept of in-silico methods permeates every early stage shown in Figure 1.1, this thesis will primarily concentrate on the lead identification phase.

Lead identification encompasses several subfields, including de-novo design, High Throughput Screening (HTS), and Virtual Screening. De-novo design is a method that uses computational techniques to design novel molecular structures from scratch based on the desired biological activity and selectivity, without relying on known compounds as references [58]. In contrast, Virtual Screening screens large chemical libraries to identify potential candidates from pre-existing molecules [60]. Furthermore, it's pivotal to distinguish between High-Throughput Screening (HTS) and High-Throughput Virtual Screening (HTVS). While both are integral tools in drug discovery, they serve distinct roles and processes. HTS is an experimental method that physically tests a vast number of compounds against a biological target in the lab to empirically determine those that elicit a desired biological effect [45]. In contrast, HTVS is a computational approach that uses algorithms to predict which compounds from a digital library might bind to or influence a target, eliminating the need for immediate physical testing. Compounds identified by HTVS as having high predicted affinity or activity are subsequently validated experimentally [60]. A detailed comparison between these two methods is presented in Table 1.1. This discussion will focus on Virtual Screening, especially High-Throughput Virtual Screening.

	HTVS	**HTS**
Type	Computational method	Experimental method
Process	Uses computer algorithms to predict the binding or activity of compounds from a digital library.	Physically tests compounds against a biological target in the lab.
Purpose	Identify potential drug candidates or active compounds without physical testing.	Empirically determine active compounds with desired biological effect.
Outcome	Compounds with predicted high affinity or activity are validated experimentally.	Active compounds are further refined for potential drug development.

Table 1.1: Comparison between High-Throughput Virtual Screening (HTVS) and High-Throughput Screening (HTS) [45, 60].

1.2. Virtual Screening

High-Throughput Virtual Screening (HTVS) has emerged as a powerful tool in drug discovery, addressing persistent challenges in High-Throughput Screening (HTS), such as low hit rates and high expenses. HTVS complements HTS and, when combined with structural biology, efficiently identifies potential drug candidates from extensive libraries tailored to specific targets, considering aspects of ligand flexibility and binding interactions [44]. This integrated approach significantly reduces drug discovery time and costs while lowering the failure rate. HTVS consists of two primary methods: Ligand-Based Virtual Screening (LBVS) and Structure-Based Virtual Screening (SBVS). The choice between SBVS and LBVS depends on the availability of physicochemical information. SBVS is preferred when structural data for the target protein is accessible. In cases with limited structural information, LBVS, especially effective when biological data exists for a large set of compounds, becomes the preferred approach [7].

1.2.1. Ligand-based virtual screening (LBVS)

Ligand-based virtual screening (LBVS) harnesses information from known ligands (molecules capable of binding to target proteins) to predict the activity of novel compounds. Within the LBVS framework, methods predominantly fall into three categories: similarity-based, pharmacophore-based, and Quantitative Structure-Activity Relationship (QSAR) based approach [7]. Of these, the **similarity-based approach** stands as both fundamental and paramount. It operates primarily on the similarity-property principle, positing that structurally related molecules likely possess similar biological activities. As highlighted by Willett, this foundational concept was first explicitly articulated by Johnson and Maggiora [34, 72]. Guided by this principle, a straightforward virtual screening strategy emerges: First, calculate the similarity between a known molecule (often termed as the reference or target structure) and each candidate molecule found within a database (typically a large chemical library). Next, rank the candidate molecules based on their computed similarities, and then conduct real screening exclusively on the top-ranked candidates [72].
In similarity-based virtual screening, it's essential to use an appropriate **descriptor** to represent the molecules and a precise **measure** to assess the similarity between the reference structure and the candidate molecules in the database. Descriptors can be categorized by their dimensionality into one-dimensional (1D), two-dimensional (2D), or three-dimensional (3D) types. While 1D descriptors capture bulk properties like molecular weight and size, 2D descriptors generate array representations by simplifying the molecular atomic information, such as through connectivity indices. Conversely, 3D descriptors rely on molecular conformations, for instance, it may take into account aspects like solvent-accessible surface area [27, 73]. On the other hand, for the similarity measure, as proposed by Willett, it consists of three components, "The representation that is used to characterize the molecules that are being compared; The weighting scheme that is used to assign differing degrees of importance to the various components of these representations; The coefficient that is used to determine the degree of relatedness between two structural representations" [72]. Since we utilized a specific 2D descriptor, a type of 2D Fingerprint known as Extended-connectivity fingerprints (ECFPs), to assess the diversity

of ligands in our dataset, we decided to use the 2D Fingerprint as a representative example of a descriptor. A detailed discussion on Extended-connectivity fingerprints (ECFPs) can be found in Section 1.4.

The **2D Fingerprint** is a computational representation of a molecule, translating its structural and chemical features into a binary or integer vector. This representation facilitates efficient molecular similarity and diversity analyses and serves as a specific type of structural depiction [57, 72]. After obtaining the 2D fingerprints of both the reference structure and the candidate molecules, the subsequent step involves computing their similarity by an appropriate measure. The Tanimoto coefficient is a widely used measure of this similarity. It is defined as the ratio of the intersection of the two fingerprints to their union, as depicted in Eq 1.1.

$$T(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{1.1}$$

In this equation, the numerator $A \cap B$ denotes the count of shared bits between the two fingerprints, while the denominator $A \cup B$ signifies the total bit count across both fingerprints. Additionally, |A| and |B| quantify the non-empty bins in sets A and B respectively [27]. For a comprehensive illustration of this process, we can refer to the study by Garcia-Hernandez et al. [27]. In their work, they employed the Extended reduced Graphs (ErG) methodology [62] to streamline the 2D representation of the compound. This process is depicted in Fig 1.2.

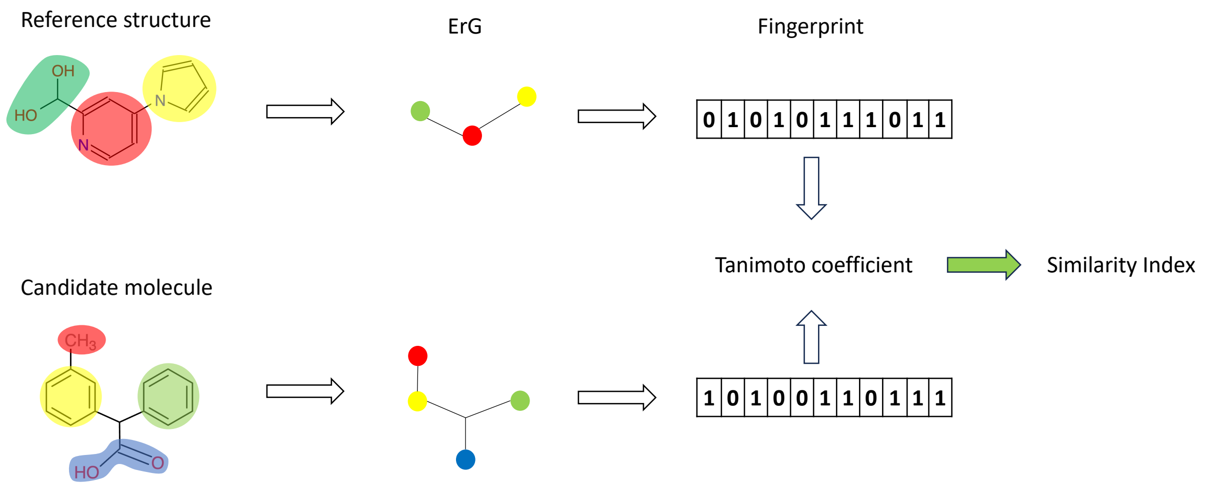


Figure 1.2: The process of calculating the Tanimoto similarity between the reference structure and the candidate molecules. [27]

As demonstrated in the previous example, the 2D fingerprint is derived from the structure of each compound. By calculating the Tanimoto coefficient between each pair of reference structure and candidate molecule, and then ranking these similarity scores, thus enabling us to efficiently execute a straightforward ligand-based virtual screening.

1.2.2. Structure-Based Virtual Screening (SBVS)

Unlike LBVS, Structure-Based Virtual Screening (SBVS) utilizes the structural information of the target protein to identify potential ligands. This approach proves especially beneficial when the structure of the target protein is known or can be reliably predicted. With the 3D structure of the target in hand, SBVS employs both docking and scoring techniques to shortlist potential candidates for in-depth analysis. The Structure-Based Virtual Screening (SBVS) workflow can be summarized into distinct phases, as depicted in Figure 1.3. It commences with the data preparation phase, during which both the compound database and the target protein undergo necessary preparations for analysis. This phase includes crucial data processing tasks such as filtering and the addition of charges to the compounds. Subsequently, the core of the SBVS process unfolds, featuring two pivotal steps: molecular docking and scoring. Molecular docking's objective is to predict the conformation and orientation (or posing) of ligands within the target binding site. This step entails comprehensive sampling of the coordinate space within the binding site, commonly known as conformational sampling or conformational searching. Following that, the scoring process evaluates each potential ligand pose, providing a predicted binding mode for each compound [38, 44]. In the post-processing phase, further operations are executed to refine the selection of potential compounds. A notable technique in this context is consensus scoring. Charifson et al. [14] demonstrated that combining the results of multiple scoring functions in an intersection-based consensus approach can significantly enhance the ability to discriminate between active and inactive enzyme inhibitors. In addition, the technique of rescoring also warrants attention. This method involves re-evaluating the top-ranked ligands from the initial screening using a more accurate scoring function, serving to refine or validate the results [56]. The ultimate step involves the selection of compounds that meet the criteria established in the earlier stages, and these chosen compounds are forwarded for experimental assay. This marks a critical transition from computational predictions to practical laboratory testing, emphasizing the iterative and interdisciplinary nature of the drug discovery process.
We will explore the details of Molecular Docking in the following Section 1.3.

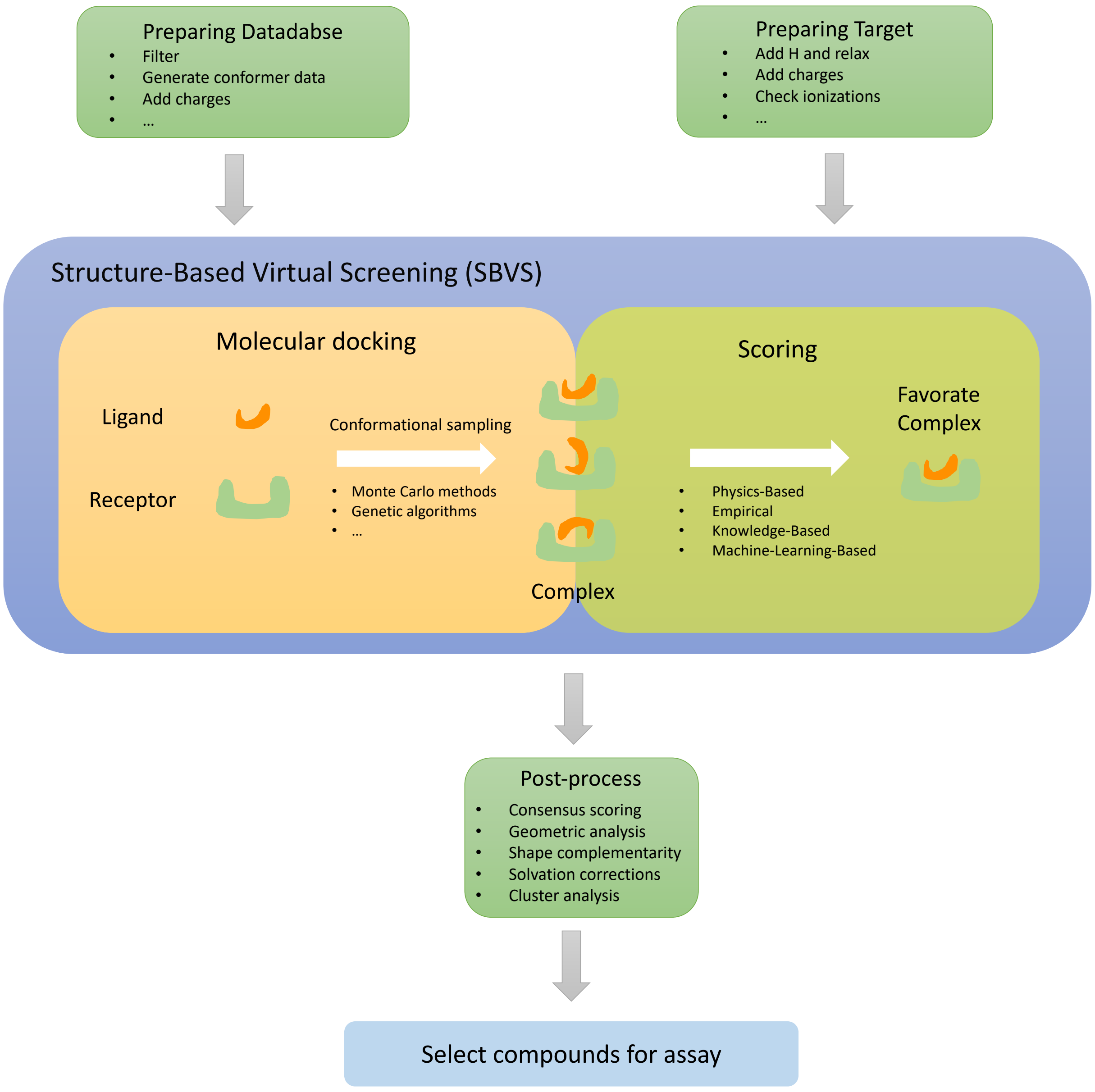


Figure 1.3: The typical workflow for virtual screening against a specific target [42, 44]

1.3. Molecular Docking

This section delves into the intricacies of molecular docking, with a primary emphasis on the conformational sampling aspect, commonly known as docking. Molecular docking is categorized into two primary approaches: rigid docking and flexible docking. To illustrate these approaches, we turn to the models developed by Fan et al. [23], presented in Figure 1.4.

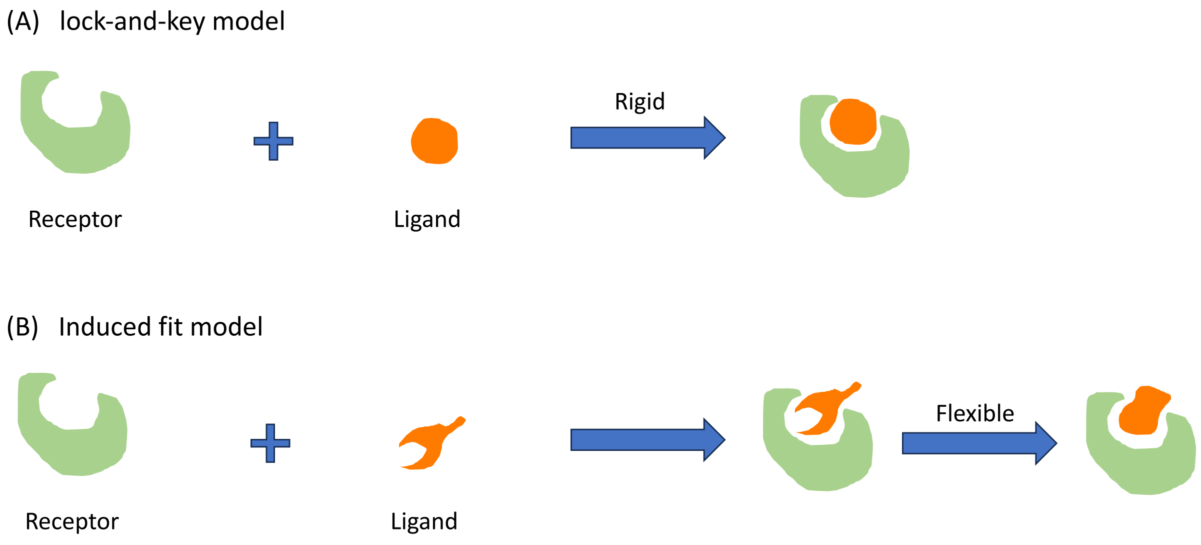


Figure 1.4: Two models of molecular docking by Fan et al. [23]

The lock-and-key model [Figure 1.4 (A)] portrays the docking of receptors and ligands as a rigid interaction, analogized to the precise fitting of a key into its corresponding lock [23, 48]. However, in reality, this process exhibits significant flexibility. This 'flexibility' denotes the capacity of receptors and ligands to dynamically adapt their shapes. Unlike static, immutable structures, these molecules have the ability to modify their conformations, similar to how a material might be bent or reshaped for a better fit. Such molecular flexibility ensures an optimized interaction between receptors and ligands. This concept aligns closely with the Induced Fit model [Figure 1.4 (B)] introduced by Fan et al. [23], where the ligand adjusts its shape to interact optimally with a specific binding site. While geometric complementarity is crucial, energy complementarity and pre-organization ensure that receptors and ligands achieve the most stable structure, ultimately minimizing free energy [5, 23, 40].
As previously discussed, the complexity of the docking process makes conformational searching especially challenging. Consequently, a reliable search algorithm becomes indispensable. Currently, various search algorithms, including Monte Carlo (MC), Genetic

Algorithms (GAs), and Simulated Annealing (SA), address both ligand flexibility and, to some extent, protein flexibility.

Category	**Method/Algorithm**
Systematic	Incremental Construction Conformational Search Databases
Random/Stochastic	Monte Carlo(MC) Genetic Algorithms Tabu Search
Simulation/Deterministic	Molecular Dynamics(MD) Energy Minimization

Table 1.2: The categories of conformational searching methods [11, 38]

These algorithms can be broadly classified into three categories [11, 38], as detailed in Table 1.2.

1. **Systematic Search Algorithms**: These algorithms utilize a grid that represents each formal 'degree of freedom'—essentially the independent parameters or directions in which a system can change. During the search, every grid value is explored combinatorially. However, as these degrees of freedom grow, the evaluations required scale up quickly. To manage this exponential growth, termination criteria are implemented to sidestep regions known to yield undesirable results.

2. **Stochastic Search Algorithms**: This method involves making random alterations, typically adjusting one degree of freedom in the system at a time. A primary challenge with stochastic searches is the 'uncertainty of convergence'. This means that there's no guaranteed path to a global minimum solution, or it might take an unpredictably long time to find it. To enhance the likelihood of convergence, multiple independent runs are typically executed.

3. **Deterministic Searches**: In this approach, the initial state determines the potential moves leading to the next state, typically ensuring that the energy of the new state is equal to or less than the starting state. A distinctive characteristic of deterministic searches is their repeatability: given the exact same starting system (including every degree of freedom) and identical parameters, they will always yield

the same final state. However, a notable limitation of deterministic algorithms is their tendency to get trapped in local minima because they cannot overcome energy barriers.

Summing up, every algorithm brings its distinct strengths and weaknesses. Various docking programs employ them based on their specific needs. Notably, certain programs may utilize multiple algorithms. For example, AutoDock might use both the Monte Carlo (MC) and Simulation methods, as well as Dock might combine Incremental Construction with the Simulation method [38].

1.4. Extended-connectivity fingerprints (ECFPs)

Extended-connectivity fingerprints (ECFPs) are a type of topological fingerprint utilized for molecular characterization. Developed specifically for structure-activity modeling, they diverge from the traditional topological fingerprints that were primarily designed for substructure and similarity searching [57].
The process of generating ECFP involves three primary stages [57]:

1. **Initial Assignment Stage:** Each atom in the molecule is allocated an integer identifier. This could be based on attributes like the atomic number.
2. **Iterative Updating Stage:**
   - Atoms collect their own identifiers and those of their immediate neighbors into an ordered array. The order is based on the identifiers and the bonds that connect them to ensure consistency(avoid order-dependence).
   - A hash function is used to transform this array into a new integer identifier for the atom.
   - These new identifiers replace the old ones, and they are added to the fingerprint set.
   - This updating process is repeated for a predefined number of iterations.
3. **Duplicate Identifier Removal Stage:** All duplicate identifiers within the set are removed. The final set of unique integer identifiers constitutes the ECFP fingerprint.

As an example of this process, Figure 1.5 illustrates the generation of the ECFP fingerprint for the molecule butyramide, showcasing three iterations: ECFP 0, ECFP 2, and ECFP 4, where the numbers 0, 2, and 4 indicate the respective search diameters. Within this figure, the three blue boxes depict the search results corresponding to these different radii. Specifically, at a search radius of 0, the sequence from left to right represents the search results for all atoms of butyramide, with ‘A’ denoting any unknown atom at this stage. Notably, the yellow areas in the figure highlight the typical duplicates that need to be eliminated. Following the removal of these duplicates, a search using the next radius is conducted for each unique structure, consistent with the methodology described in the previous paragraph. The ultimate goal of this process is to derive the ECFP fingerprint, which consists of an array consolidating unique hash values from all iterations, as exemplified by the ‘hash values’ field on the right side of the figure.

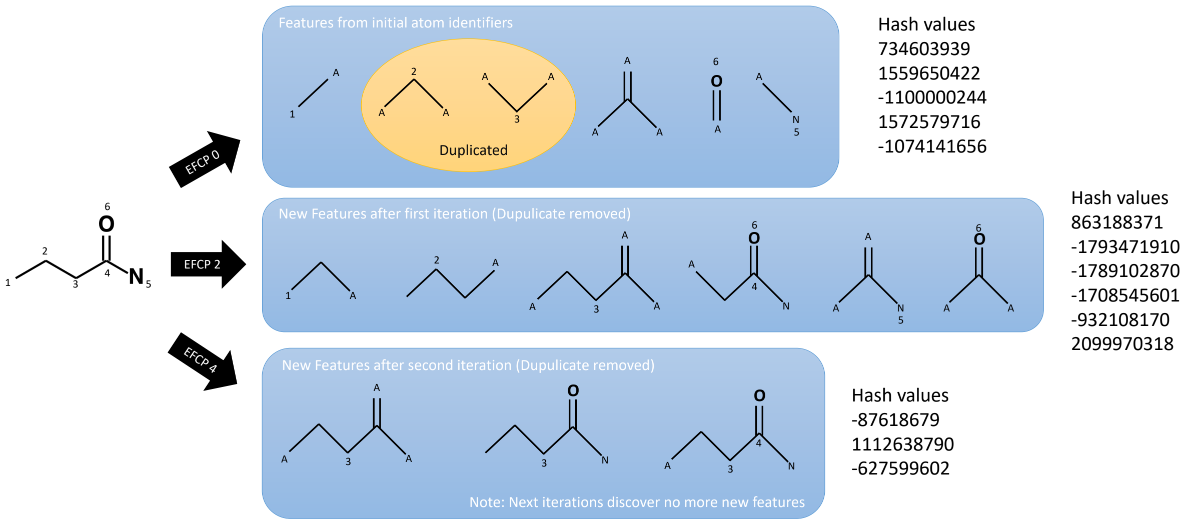


Figure 1.5: The ECFP generating process [57]

The strength of this methodology, rooted in the Morgan algorithm-based updating process [47], lies in its ability to create identifiers for large substructures using only local operations. Although each atom only interacts with its direct neighbors, the resulting identifiers can signify extensive substructures. When applied across the entire molecule, the resulting set encapsulates substructural information from every part, providing a blend of detailed and generalized structural information [57].
ECFPs come with several advantageous qualities:

- Rapid calculation capability.
- Non-predefined nature, allowing representation of a vast array of molecular features, inclusive of stereochemical information.
- Features that signify the presence of specific substructures, aiding in the clearer interpretation of analysis outcomes.
- The adaptability of the ECFP algorithm, which can be tweaked to produce varying

kinds of circular fingerprints suitable for diverse applications.

ECFPs have been applied in a wide range of applications, including Structure-Activity Relationship Modeling, Virtual Screening, and Compound Library Analysis [57]. In our methodology, assessing the diversity of ligands in the database is crucial. We employed ECFPs to produce molecular descriptors and then computed the Tanimoto coefficients for each pair of ligands. This quantitative analysis of ligand diversity allowed us to judiciously exclude overly similar ligands from the training set, mitigating potential model bias.

1.5. PDBbind database

PDBbind [53] is a comprehensive biomolecular database offering a curated collection of experimentally measured binding affinity data for biomolecular complexes sourced from the Protein Data Bank (PDB). Between 2015 to 2020, PDBbind consistently updated its database in response to the ongoing expansion of PDB. As of 2023, the latest available version is PDBbind v2020. This version encompasses various complexes, including protein-ligand, nucleic acid-ligand, protein-nucleic acid, and protein-protein. In our methodology, we exclusively used the protein-ligand complexes. These are further subdivided into two sets: the "refined set", consisting of 5,316 complexes, and the "general set", which includes 14,127 complexes.

For every protein-ligand complex, PDBbind provides PDB files detailing both the entire protein and its distinct binding pocket. The protein's PDB file captures comprehensive atomic information, including atoms' 3D coordinates, bond types, residue names, and more. Meanwhile, the pocket represents a specialized area on the protein's surface where molecules, especially ligands, can bind, influencing the protein's function. Ligands are presented in both mol2 and sdf formats. Both formats store molecular structure data; however, the mol2 format offers additional details like SYBYL atom types, making it particularly useful when its atom types assignment quality can be trusted.

In the PDBbind database, the most crucial information for our analysis of protein-ligand complexes is the binding affinity data, which is provided in the forms of K_d, K_i, and IC_{50}. Of these, the dissociation constant K_d, is an equilibrium constant signifying the tendency of a larger entity to reversibly disassemble into its smaller components. It's frequently employed to describe the binding affinity between a ligand L and a protein P, indicating how tightly the ligand associates with the receptor [12]. The interaction between a ligand and protein can be conceptualized as a two-state process. As illustrated in Equation 1.2, the ligand-protein complex can revert to its separate entities L and P and vice versa, maintaining a dynamic equilibrium upon reaching a stable status.

$$LP \rightleftharpoons L + P \tag{1.2}$$

The corresponding dissociation constant (K_d) is defined in Equation 1.3. Within this equation, $[L]$ and $[P]$ signify the concentrations of free ligand and protein, respectively, whereas $[LP]$ indicates the concentration of the ligand-protein complex when 50% of the receptors are bound by ligands. A smaller K_d value signifies a tighter ligand binding, reflecting a higher affinity between the ligand and receptor.

$$K_d = \frac{[L][P]}{[LP]} \tag{1.3}$$

The inhibitory constant (K_i) is a distinct form of equilibrium dissociation constant (K_d), signifying the equilibrium binding affinity of a ligand that diminishes the activity of the molecule it binds to. K_i denotes the inhibitor ligand concentration required to occupy half of the receptor sites in the absence of any competing ligands. A lower K_i value indicates a higher binding affinity, meaning less inhibitor ligand is necessary to inhibit the activity of its binding target [12].

Additionally, PDBbind includes the Half-maximal inhibitory concentration (IC_{50}) as a measure of binding affinity for certain protein-ligand complexes. IC_{50} denotes the concentration at which a competitive antagonist decreases the activity/binding of an agonist to a specific enzyme, receptor, or transporter by half. This metric can be derived from in-vitro or ex-vivo binding assay curves and serves to quantify the antagonist's affinity for an enzyme or receptor. The IC_{50} value is influenced by three key aspects: the antagonist's receptor affinity (with higher affinity leading to a lower IC_{50}), the concentration of the displaced ligand (where greater ligand concentrations necessitate more antagonist for 50% inhibition, raising the IC_{50}), and the dissociation constant (K_d) for the ligand-receptor pairing (with a smaller K_d meaning more antagonist is required to oust the ligand, hence a higher IC_{50}) [12].

Finally, while PDBbind offers various measurements (K_d, K_i, and IC_{50}) for distinct protein-ligand complexes, they share consistent units. For our data analysis, we can consider these measurements as equivalent, or choose to use the -log transformed data directly provided by PDBbind.

1.6. The CASF-2016

CASF-2016 [63], representing the Comparative Assessment of Scoring Functions - 2016 update, is a widely recognized benchmark for evaluating scoring functions. Dramatically speaking, one might describe it as "a scoring function for scoring functions". CASF-2016 provides a test dataset comprising 285 protein-ligand complexes, each marked by high-quality crystal structures and reliable binding constants. This benchmark is anchored by four critical metrics - scoring power, ranking power, docking power, and screening power - each designed to objectively discern the strengths and weaknesses of a particular scoring function. We will delve deeper into these four evaluation metrics in this section.

Scoring Power evaluates the proficiency of a scoring function in producing binding scores that have a linear correlation with the experimental values of protein-ligand complexes. The primary metric utilized is the Pearson correlation coefficient (R), as illustrated in Equation 1.4, comparing predicted binding affinities with their experimental counterparts. In this equation, x_i represents the binding score for the ith complex, $\bar{x}$ is the average score, and y_i denotes the experimental binding constant for that complex. A higher Pearson correlation coefficient denotes greater accuracy in scoring power.

$$R = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i^n (x_i - \bar{x})^2}\sqrt{\sum_i^n (y_i - \bar{y})^2}} \tag{1.4}$$

As a reference, CASF-2016 also documents the standard deviation (SD) between the experimental and predicted values, as seen in Equation 1.5. Here, the predicted values undergo linear regression, with "a" and "b" denoting the intercept and slope of the regression line, respectively.

$$SD = \sqrt{\frac{\sum_i^n [y_i - (a + bx_i)]^2}{n-1}} \tag{1.5}$$

Ranking Power measures a scoring function's proficiency in accurately ordering the known ligands of a specific target protein based on their binding affinities, given the exact binding poses. Unlike scoring power, ranking power doesn't necessitate a linear relationship between calculated binding scores and experimental binding results. The primary metric employed is the Spearman's rank correlation coefficient (ρ), detailed in Equation 1.6. In this context, rx_i represents the rank of the binding score for the ith complex, ry_i denotes the rank of its experimental binding constant, and n is the total sample count.

$$\rho = \frac{\sum_i^n (rx_i - \bar{rx})(ry_i - \bar{ry})}{\sqrt{\sum_i^n (rx_i - \bar{rx})^2}\sqrt{\sum_i^n (ry_i - \bar{ry})^2}} \tag{1.6}$$

CASF-2016 also employs Kendall's rank correlation coefficient to assess the Ranking Power, as detailed in Equation 1.7. In this equation, $P_{concord}$ and $P_{discord}$ denote the count of matching and mismatching pairs, respectively. Each sample i is defined by (x_i, y_i), where x_i is the calculated binding data and y_i is its observed counterpart. A pair (i, j) matches if both x and y rankings align (either $x_i > x_j$ with $y_i > y_j$ or vice versa). If they contradict, they're mismatched. T and U are the counts of tied rankings for x and y. If $x_i = x_j$ and $y_i = y_j$ simultaneously, that pair isn't included in T and U. The denominator represents the total possible pair combinations among the samples.

$$\tau = \frac{P_{concord} - P_{discord}}{\sqrt{(P_{concord} + P_{discord} + T)(P_{concord} + P_{discord} + U)}} \tag{1.7}$$

Additionally, the Predictive Index (PI) [54] is also adopted by CASF-2016, as detailed in Equation 1.8. In this metric, W_{ij} denotes the difference in experimental binding data between ligand i and j, given as $W_{ij} = abs\,(E_j - E_i)$. C_{ij} evaluates the alignment between the rank orders of experimental binding data (E_i and E_j) and predicted scores (P_i and P_j). Specifically, $C_{ij} = 1$ when $\frac{E_j - E_i}{P_j - P_i} > 0$, $C_{ij} = -1$ when $\frac{E_j - E_i}{P_j - P_i} < 0$, and $C_{ij} = 0$ if they are equal. Beyond its basic ranking capability, PI underscores the significance of correctly ranking ligands with marked differences in experimental binding data.

$$PI = \frac{\sum_{j>i}^n \sum_i^n W_{ij} C_{ij}}{\sum_{j>i}^n \sum_i^n W_{ij}} \tag{1.8}$$

Docking Power measures a scoring function's proficiency in distinguishing the native ligand binding pose from computer-generated decoys. As depicted in Figure 1.6, the scoring function under evaluation is tasked with scoring ligand-docked complexes. All these complexes feature the same protein/ligand but in various docked poses and positions. Among the set, one represents the native ligand binding pose, while the rest are computer-generated decoys. Once scored, the complexes are ranked based on their scores. Subsequently, the similarity between the native binding pose and the top-ranked binding pose is quantified using Root Mean Square Deviation (RMSD) values. This RMSD metric, introduced by Allen et al. [4], is computed based on the Hungarian algorithm, as

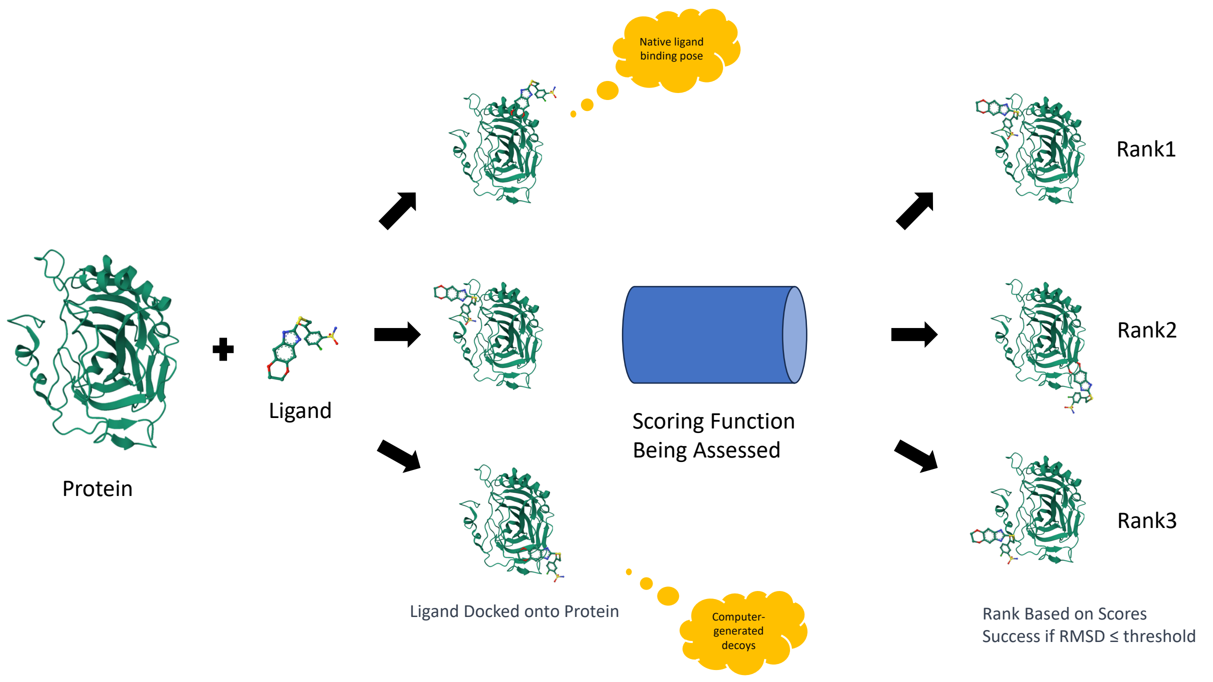

Native ligand binding pose
Protein
Ligand
Ligand Docked onto Protein
Computer-generated decoys
Scoring Function Being Assessed
Rank1
Rank2
Rank3
Rank Based on Scores
Success if RMSD ≤ threshold

Figure 1.6: Docking Power assessment process

detailed in Equation 1.9.

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left((x_i - x_i')^2 + (y_i - y_i')^2 + (z_i - z_i')^2\right)} \tag{1.9}$$

In this equation, N represents the total number of heavy atoms in the ligand molecule. The coordinates x_i, y_i, and z_i correspond to the ith atom in the native binding pose, while x_i', y_i', and z_i' refer to its coordinates in the top-ranked binding pose. The final Docking Power evaluation results are indicated by success rates of ranking Top1, Top2, and Top3. A docking is deemed successful when its RMSD value falls below a set threshold, typically 2.0 Å.

Screening Power measures a scoring function's proficiency in identifying the true binder for a target protein amidst a collection of random ligands with varied binding poses. In CASF-2016, the screening power was assessed using a cross-docking approach. As illustrated in Figure 1.7, each target protein has 285 associated ligands: 5 positives and 280 negatives. For every ligand, 100 potential binding poses are prepared. The scoring function under evaluation needs to score all these protein-ligand pairs across their varied poses. The highest binding score among the poses is noted as the predicted score for that ligand. Subsequently, all 285 ligands are ranked in descending order based on these scores. The scoring function's screening power is determined by its ability to rank the

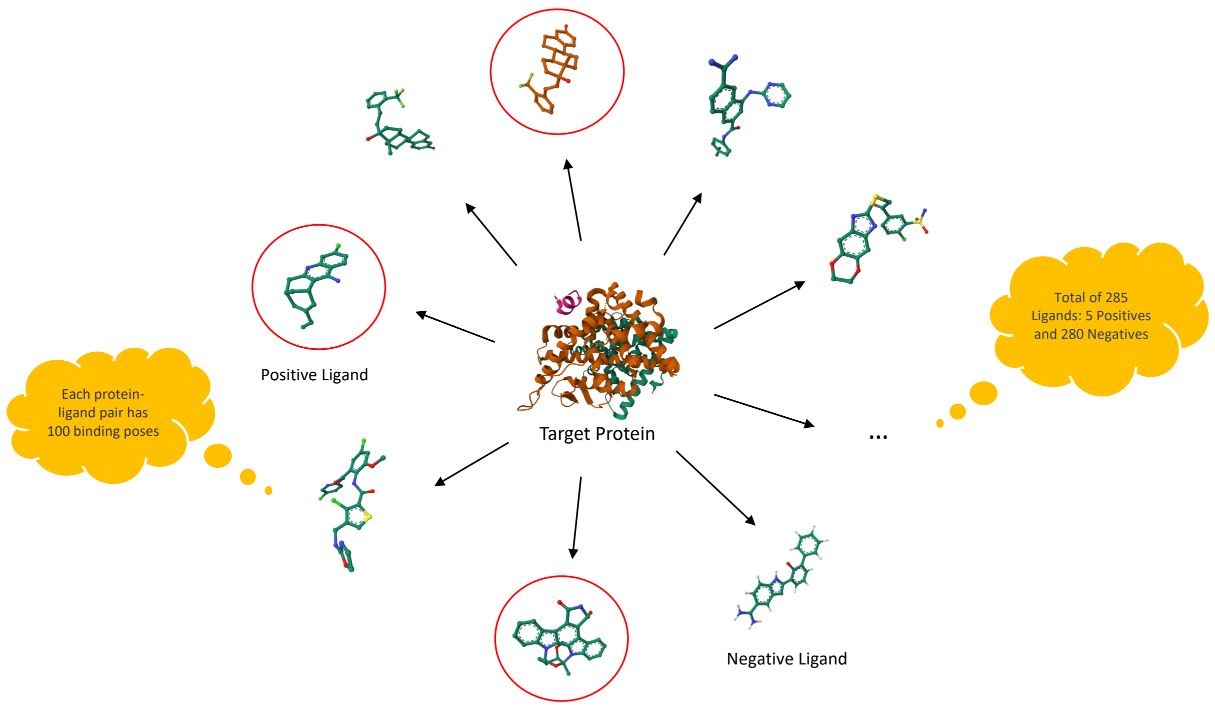

Positive Ligand
Target Protein
Negative Ligand
Each protein-ligand pair has 100 binding poses
Total of 285 Ligands: 5 Positives and 280 Negatives
...

Figure 1.7: Explanation of the Screening Power Test Set

true binders (positives) towards the top positions.
Unlike other versions of CASF, CASF-2016 splits Screening Power into two distinct parts: forward and reverse screening power. The forward screening power aims to identify potential small-molecule ligands for a selected target protein. Its criteria include both the success rate, similar to docking power, and the enhancement factor (EF) which is unique to the forward screening power test. The success rate indicates the accuracy of identifying the true positive binder by assessing if the best ligand is found within the top 1%, 5%, or 10% of candidates across all 57 target proteins in the test set. Also, the enhancement factor, defined in Equation 1.10, quantifies the enrichment of true binders among the top-ranked ligands relative to a random ranking expectation. In this equation, NTB_α denotes the number of true binders observed among the top α-ranked candidates (with α being 1%, 5%, or 10%) selected by the scoring function being assessed. NTB_α represents the total number of true binders for a given target protein, which is five in CASF-2016. Lastly, the enhancement factor is averaged across all 57 target proteins in the dataset.

$$\mathrm{EF}_\alpha = \frac{\mathrm{NTB}_\alpha}{\mathrm{NTB}_{\mathrm{total}} \times \alpha} \tag{1.10}$$

On the other hand, the reverse screening power focuses on identifying potential target proteins for a given ligand. This is a new feature in CASF and is simple, requiring no extra computations. Unlike the forward screening power, which ranks ligands for each protein, the reverse screening power ranks proteins for each ligand. The indicator for reverse screening power is the success rate, determined by whether the optimal target is found within the top 1%, 5%, or 10% of rankings.

1.7. High Performance Computing

In this section, we present a hierarchical model for introducing High-Performance Computing (HPC), as illustrated in Figure 1.8. This model systematically categorizes HPC into distinct tiers, offering a logical and structured overview. At the broadest tier, the **Computer infrastructure level**, there's an emphasis on leveraging large-scale infrastructures, exemplified by supercomputers. Moving to the **Computer hardware level**, the spotlight is on exploiting parallel computing capabilities through technologies such as multi-core CPUs, GPUs, FPGAs and specially-designed hardware accelerators. At the **Computing framework level**, the focus is on incorporating parallel computing frameworks like MPI, OpenCL, and CUDA. Finally, at the **Computing algorithm level**, the essence is on devising high-performance algorithms tailored to specific computational tasks.

The levels mentioned above are set to interplay and complement one another. For instance, when developing a high-performance algorithm for a specific task, it is essential to consider not only the task's execution, time complexity, and space complexity but also the effective use of the computing framework. This includes invoking tools like OpenMP for multithreading and CUDA for GPU acceleration, ensuring seamless alignment with the computing hardware and even underlying infrastructure, such as supercomputer, to maximize computational potential.

1.7.1. Computer infrastructure level

Today, using large-scale cluster computer infrastructure for high-performance computing has become commonplace. As of June 2023, according to the 61st edition of the TOP500 report, the world's fastest supercomputer is "Frontier" at Oak Ridge National Laboratory in the U.S. It boasts 8,699,904 cores and its theoretical peak performance reaches an impressive 1,679.82 Peta Floating Point Operations Per Second (PFLOPS) [66].

Supercomputer architectures are sophisticated, encompassing a multitude of diverse components. Consider the 'Node Architecture', Even back in 2004, the supercomputer "BlueGene/L" of IBM already boasted 65,536 nodes [2]. Each node in such setups functions akin to a standalone computer, typically equipped with multiple CPUs or GPUs, memory, and sometimes storage. Given the sheer number of nodes, the role of 'Interconnects'

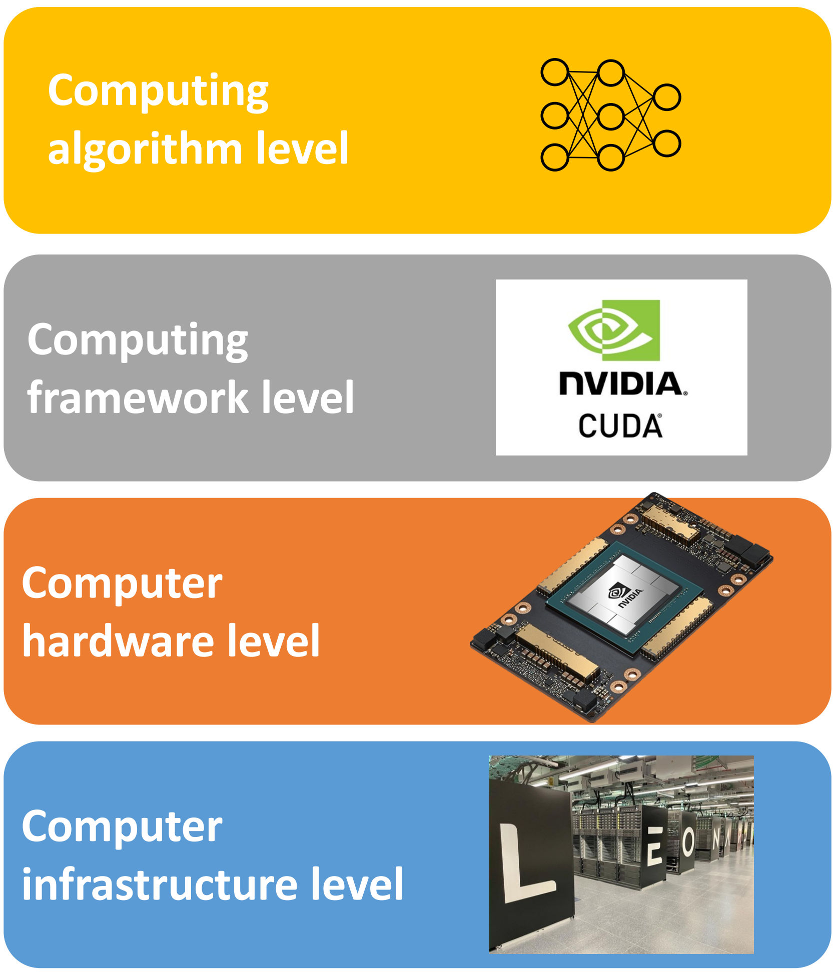

Computing algorithm level
Computing framework level
NVIDIA
CUDA
Computer hardware level
Computer infrastructure level

Figure 1.8: A Hierarchical Model for Introducing High-Performance Computing (HPC)

becomes paramount. Nodes within a supercomputer are interconnected, predominantly via high-speed networks. As a case in point, Italy's supercomputer "Leonardo" employs a Quad-rail NVIDIA HDR100 Infiniband for its interconnection, which can transfer 200 Gb of data per second [17]. This interconnection component enables super rapid data transfers between nodes, crucial for tasks necessitating regular inter-node communication. Beyond this, the integrity of the entire system often relies on other essential components, ranging from Storage clusters to the likes of Cooling Systems, Fault Tolerance mechanisms, Redundancy strategies, and dedicated Power systems.

Using such large-scale computing infrastructures in drug discovery is becoming increasingly prevalent. For example, Gadioli et al. [26] utilized the EXSCALATE platform on two HPC machines: ENI HPC5 and CINECA Marconi100, ranked 15th and 26th on the 61st edition of TOP500 [66], respectively. After combining them, the two machines delivered a peak throughput of 81 PFLOPS. They processed a chemical library with over 70 billion ligands, targeting 15 binding sites across 12 SARS-CoV-2 viral proteins. Astonishingly, in a span of just 60 hours, they achieved the colossal feat of executing a TRILLION dockings. However, even such powerful, expansive large-scale infrastructures have their limitations. Specifically, they typically serve as public computing resources and possess inherent barriers that not all researchers can easily navigate. Thus, it's essential to acknowledge the significance of compact, cost-effective, and readily accessible computing hardware for high-performance computing tasks.

1.7.2. Computer hardware level

In the realm of the computer hardware level for high-performance computing, the emphasis is squarely on parallel computing, primarily facilitated by multi-core CPUs/GPUs. Additionally, this level incorporates specialized hardware accelerators tailored for distinct computational tasks. To gain a deeper understanding, it's essential to familiarize ourselves with three foundational theoretical concepts: Instruction-Level Parallelism, Data-Level Parallelism, and Thread-Level Parallelism.
Since approximately 1985, processors have utilized pipelining to overlap the execution of instructions, enhancing performance. This concept of overlapping instructions is termed **Instruction-level parallelism (ILP)**, as it allows for parallel evaluation of the instructions. Two primary strategies exist for leveraging ILP: (1) a hardware-centric approach that dynamically identifies and utilizes parallelism, and (2) a software-driven method that seeks out parallelism statically during compilation. Processors adopting the dynamic, hardware-driven technique, like the Intel Core series, are predominant in the desktop and server sectors [30, p.148]. The interest in ILP reached its zenith at the start of the 21st century, marked by the emergence of iconic processors like the Pentium 4. This processor employed speculative scheduling with seven functional units and featured a pipeline exceeding 20 stages in depth [30, p.245]. While instructions can overlap and execute in parallel to some extent, there are inherent limitations and challenges to this approach. A principal barrier to parallel execution is the emergence of "hazards," which can be categorized into three types: [52, p.277 - p.284]:

1. **Structural Hazard**: This arises when an intended instruction cannot proceed in its designated clock cycle due to hardware constraints that prevent the simultaneous execution of a specific combination of instructions.

2. **Data Hazard (Pipeline Data Hazard)**: Occurs when an instruction cannot execute in its assigned clock cycle because the necessary data is not yet available.

3. **Control Hazard (Branch Hazard)**: This takes place when the fetched instruction isn't the one required, meaning the sequence of instruction addresses doesn't match

the pipeline's expectations.

Indeed, numerous technologies have been developed to mitigate these hazards. For example, to address data hazards, techniques like forwarding (or bypassing) and pipeline stalling (often referred to as "bubbling") are employed. For control hazards, branch prediction is a common solution. However, diminishing returns are evident in this domain as well. While some challenges can be addressed, inherent boundaries persist. These limitations have steered interest towards multicore architectures. Nonetheless, understanding these limitations is essential in striking a balance between ILP and thread-level parallelism [30].
Shifting our attention to **Data-Level Parallelism (DLP)**, it's essential to mention a classification of parallel hardware proposed in the 1960s that remains relevant today. This classification hinges on the number of instruction streams and data streams. Specifically [52]:

- **SISD** (Single Instruction stream, Single Data stream): The Instruction-level parallelism discussed earlier primarily focuses on SISD, which is a characteristic of a standard uniprocessor.
- **SIMD** (Single Instruction stream, Multiple Data streams): Here, the same instruction operates on multiple data streams, typical in a vector processor.
- **MISD** (Multiple Instruction streams, Single Data stream): A potential example of an MISD processor could be a "stream processor". This would involve executing a series of computations on a single data stream in a pipeline manner, such as parsing network input, decrypting and decompressing the data, searching for matches, and more.
- **MIMD** (Multiple Instruction streams, Multiple Data streams): Typically associated with standard multiprocessors, like CPUs and GPUs.

In the context of Data-Level Parallelism, our primary focus is on SIMD, where a single instruction processes identically structured data. A quintessential application of this architecture is the Graphics Processing Unit (GPU), which is also a crucial hardware component for High-Performance Computing at the computer hardware level. It's important to note that a GPU is not strictly an SIMD processor, it contains an array of multithreaded SIMD units. Essentially, a GPU is a MIMD architecture constructed from these multithreaded SIMD processors [52, p.526]. In our methodology, we utilize Nvidia's GPU. Therefore, we'll use the Nvidia GPU as a representative example to delve into GPU architecture and memory structures.

NVIDIA offers an array of specialized **GPU architectures**, including Fermi, Kepler, Maxwell, Pascal, Volta, Turing, Ampere and Hopper. Among these, while each possesses its distinct performance attributes, we will primarily delve into Fermi. NVIDIA has segmented the Fermi architecture into four models, each tailored for a different price point, and equipped with either 7, 11, 14, or 15 multithreaded SIMD processors. To guarantee consistent scalability across different GPU models, the Thread Block Scheduler hardware is designed to assign thread blocks to the appropriate SIMD processors. A streamlined depiction of the datapath within a multithreaded SIMD processor is illustrated in Figure 1.9 [52, p.526]. This particular processor showcases 16 SIMD lanes, and its SIMD Thread Scheduler operates multiple independent SIMD threads, selecting which ones to run on the processor.

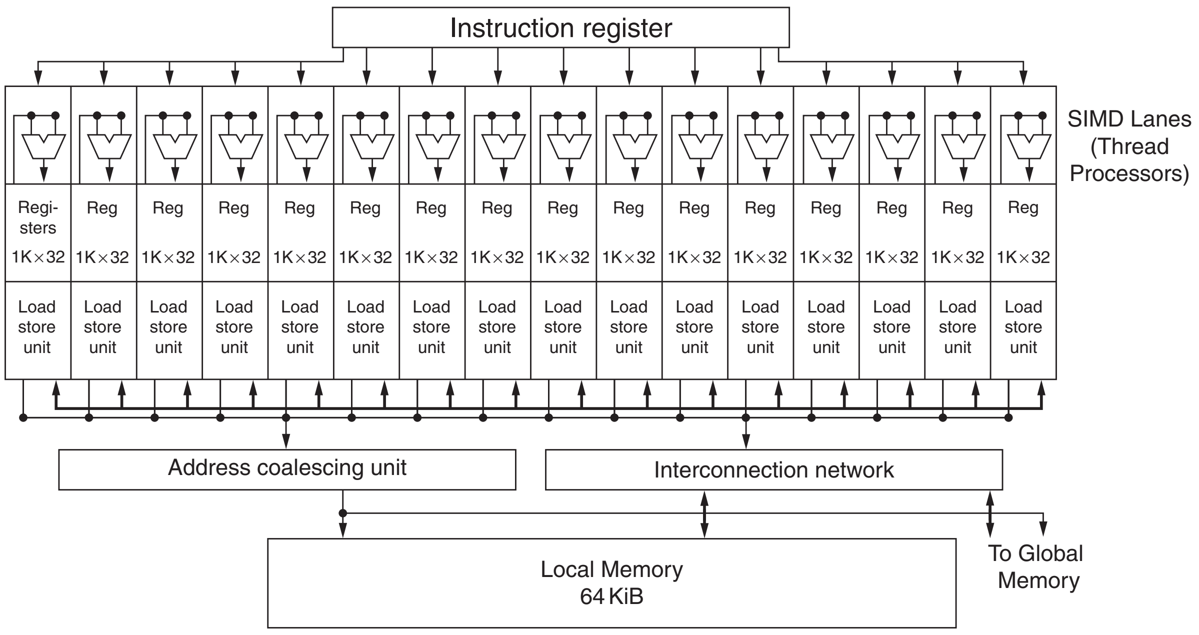


Figure 1.9: A simplified representation of the datapath of a multithreaded SIMD processor by Patterson and Hennessy [52, p.526]

In GPU architecture, **memory structures** also hold significant importance. Thoroughly understanding the memory structure of the GPU you're working with is essential for optimal high-performance programming. Within NVIDIA GPUs, each multithreaded SIMD processor is equipped with an on-chip memory known as Local Memory. This memory is shared among the SIMD Lanes of that specific processor but is not shared with other multithreaded SIMD processors. The entire GPU, which encompasses all thread blocks, taps into the off-chip DRAM, often termed GPU Memory. As depicted in Figure 1.10, the memory hierarchy of NVIDIA GPUs can be further detailed: each thread maintains its own private memory, and every thread block has local memory. Threads executing

SIMD instructions pool resources from this Local Memory. Additionally, the entire GPU Memory, covering all thread blocks, is open for access to vectorized loops. [52, p.528]. Please note that the memory terminology used here, such as 'Local memory', reflects Patterson et al.'s hardware-centric interpretation. This might differ from CUDA's official definitions from the computing framework perspective. For CUDA's definitions, refer to the subsection 1.7.3.

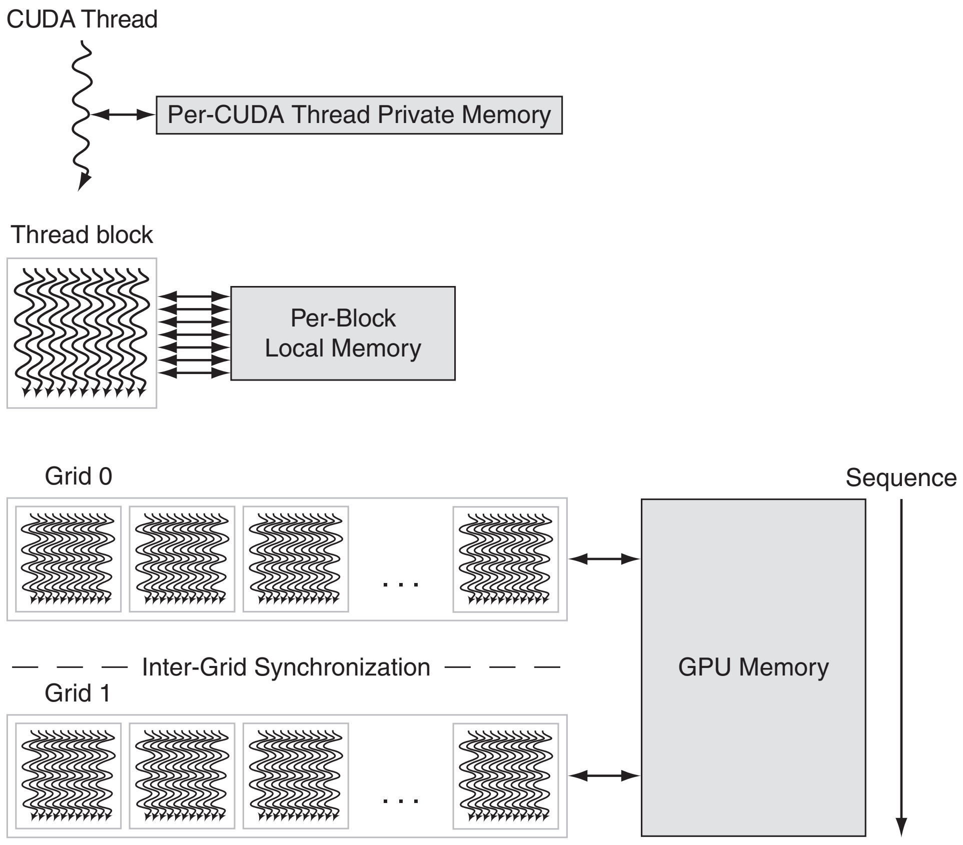


Figure 1.10: The GPU Memory structures by Patterson and Hennessy [52, p.528]

Lastly, we delve into **Thread-Level Parallelism (TLP)**. As previously noted, the emphasis on TLP has grown mainly because of the stagnation in uniprocessor performance. This stagnation results from the diminishing benefits of instruction-level parallelism (ILP) and increasing concerns about power consumption. TLP heralds a transformative period in computer architecture, marking the rise of multiprocessors from entry-level to high-end systems. TLP inherently involves multiple program counters and is predominantly leveraged through MIMD. Our discussion centers on multiprocessors, defined as computers with closely integrated processors. These processors, under the coordination of a singular operating system, share memory within a unified address space [30, p.344, p.345]. Shared-memory multiprocessors are categorized into two groups based on the number of

processors and the resulting memory organization.
The first type, as shown in Figure 1.11, known as symmetric shared-memory multiprocessors (SMPs), usually has up to eight cores. These processors equally access a centralized memory, hence the name 'symmetric'. SMP systems are also referred to as Uniform Memory Access (UMA) multiprocessors because all processors access memory at the same speed, regardless of multiple memory banks. Notably, while all cores in a multicore system share a central memory, in configurations where multiple multicore systems are connected, each system has its own memory, resulting in a distributed memory architecture.

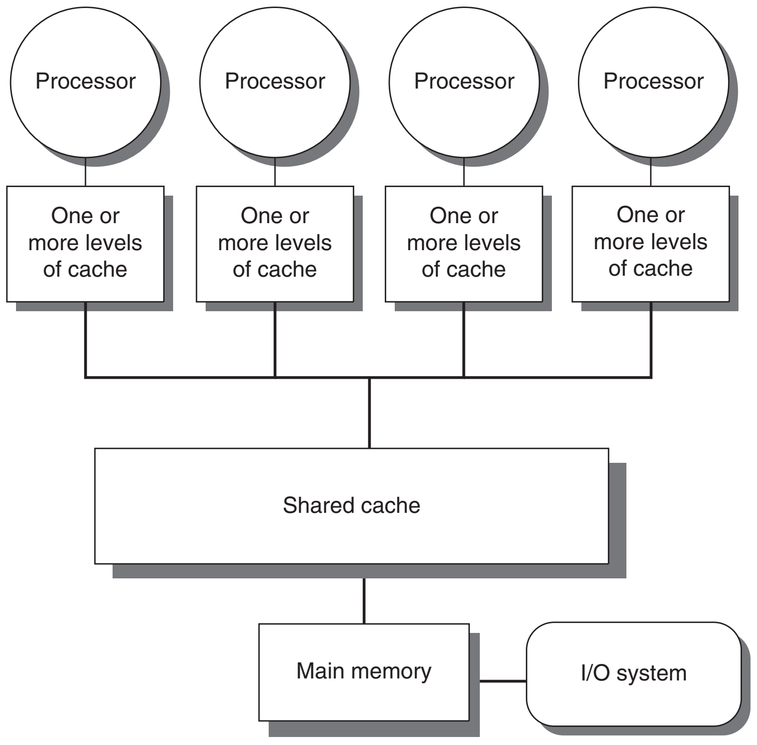

Processor
Processor
Processor
Processor
One or more levels of cache
One or more levels of cache
One or more levels of cache
One or more levels of cache
Private caches
Shared cache
Main memory
I/O system

Figure 1.11: Symmetric Shared-memory Multiprocessors (SMPs) by Hennessy and Patterson [30, p.347]

Another design method involves multiprocessors with distributed memory, known as distributed shared memory (DSM). As shown in Figure 1.12, to accommodate more processors, memory is distributed instead of centralized. This is because a central memory system can't handle the bandwidth needs of numerous processors without causing significant access delays. As processors become faster with heightened memory bandwidth demands, distributed memory is preferred, not just for larger multiprocessors but also for configurations as compact as two-chip multicore processors. It's in this context that DSM multiprocessors, also known as NUMA (nonuniform memory access), have varying access times based on data's location in memory. While the DSM's distribution of memory across nodes enhances bandwidth and minimizes local memory latency, it introduces complexity in data communication between processors. Moreover, DSM demands additional software effort to leverage the augmented memory bandwidth offered by its distributed structure [30, p.347, p.348]. Regardless of whether it's an SMP or DSM setup, processors communicate through a shared address space. This means any processor can access any memory location with the right permissions. Both SMP and DSM have 'shared memory' because of this communal address space [30, p.348].

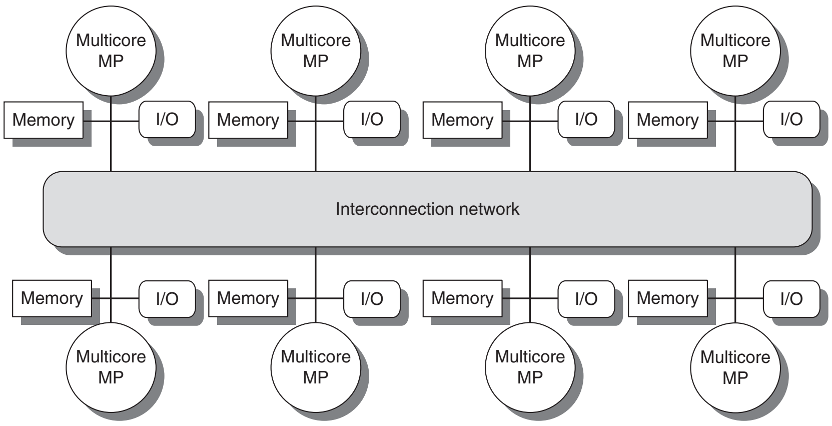


Figure 1.12: Distributed Shared-memory Multiprocessors (DMPs) by Hennessy and Patterson [30, p.348]

1.7.3. Computing framework level

Several parallel computing APIs and frameworks are designed to drive your devices for high-performance computing tasks. To offer a structured overview, we've categorized them by their main focus: CPUs or GPUs, as detailed in Table 1.3. However, the demarcation between these tools isn't always distinct. Many are multifaceted and can be applied beyond their core intention. For example, Intel's oneAPI encompasses both CPUs and GPUs but notably gravitates towards GPU operations [33]. Likewise, while Apple's Metal is equipped to manage tasks across a spectrum of processors, it predominantly emphasizes GPU-centric activities [32]

In our methodology, we leveraged **CUDA (Compute Unified Device Architecture)** to drive our high-performance computing hardware - the NVIDIA A-100 GPU. Therefore, this section will center on CUDA, showcasing it as a quintessential example of parallel computing frameworks. NVIDIA introduced CUDA in 2006 as a general-purpose parallel computing platform and programming model, specifically crafted to tap into the parallel compute engine of NVIDIA GPUs. While CUDA's software environment predominantly supports high-level programming in C++, it is versatile enough to accommodate other

Parallel Computing APIs/Frameworks	
Target Hardware	**APIs/Frameworks**
Primarily for CPU	OpenMP (Open Multi-Processing) pthreads (POSIX Threads) MPI (Message Passing Interface) oneTBB (oneAPI Threading Building Blocks) Cilk, Cilk++, Cilk Plus and OpenCilk
Primarily for GPU	CUDA (Compute Unified Device Architecture) AMD ROCm Microsoft DirectCompute
Both CPU and GPU (Heterogeneous Computing)	OpenCL (Open Computing Language) SYCL HIP (Heterogeneous-Compute Interface for Portability) Intel oneAPI Metal (by Apple)

Table 1.3: Classification of Parallel Computing APIs/Frameworks

languages, APIs, and directives-based approaches like FORTRAN, DirectCompute, and OpenACC [18, Chapter 1.2].

One of CUDA's defining features is its scalability. Given the significant variation in core numbers across NVIDIA's product range — from the affordable GeForce GPUs to the professional Quadro and Tesla units — a robust scale parallel programming model is essential. CUDA addresses this need through three foundational abstractions: a hierarchy of thread groups, shared memories, and barrier synchronization. As described by the CUDA official introduction [18, Chapter 1.3]:

> "These abstractions provide fine-grained data parallelism and thread parallelism, nested within coarse-grained data parallelism and task parallelism. They guide the programmer to partition the problem into coarse sub-problems that can be solved independently in parallel by blocks of threads, and each sub-problem into finer pieces that can be solved cooperatively in parallel by all threads within the block."

The decomposition described above not only preserves language expressiveness by facilitating thread cooperation during the resolution of each sub-problem but also ensures "Automatic Scalability". As illustrated in Figure 1.13, a multithreaded CUDA program is depicted as divided into various blocks. Different GPU models feature varying numbers of Streaming Multiprocessors (SMs). To run a compiled CUDA program, only the count of physical multiprocessors is needed. Notably, each block of threads can be scheduled to any available multiprocessor within a GPU, either concurrently or sequentially, and in any order [18, Chapter 1.3].

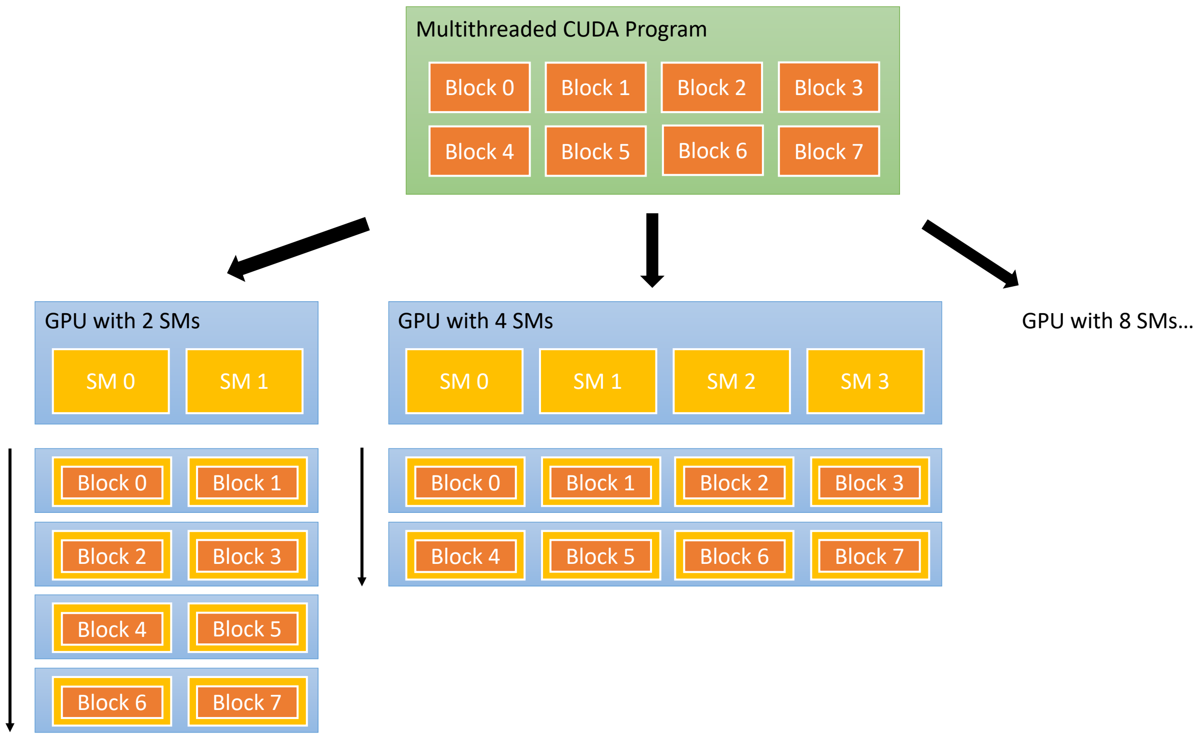


Figure 1.13: Automatic Scalability [18, Chapter 1.3]

An integral aspect of CUDA's parallel computing model lies in its distinct approach to structuring computational tasks and data management: the Thread Hierarchy and Memory Hierarchy. These hierarchies underpin the framework's capacity to efficiently distribute workloads across the vast parallel architecture of NVIDIA GPUs. By understanding these hierarchies, developers can harness the full potential of CUDA, optimizing performance by tailoring computation to the GPU's unique characteristics. As depicted in Figure 1.14, threads perform parallel computations in CUDA and are grouped into blocks. These blocks form a grid. Each block has its own shared memory accessible to its threads, streamlining inter-thread communication within that block. On the other hand, all threads, irrespective of their block, can access the global memory; however, this comes at a higher latency than the shared memory.

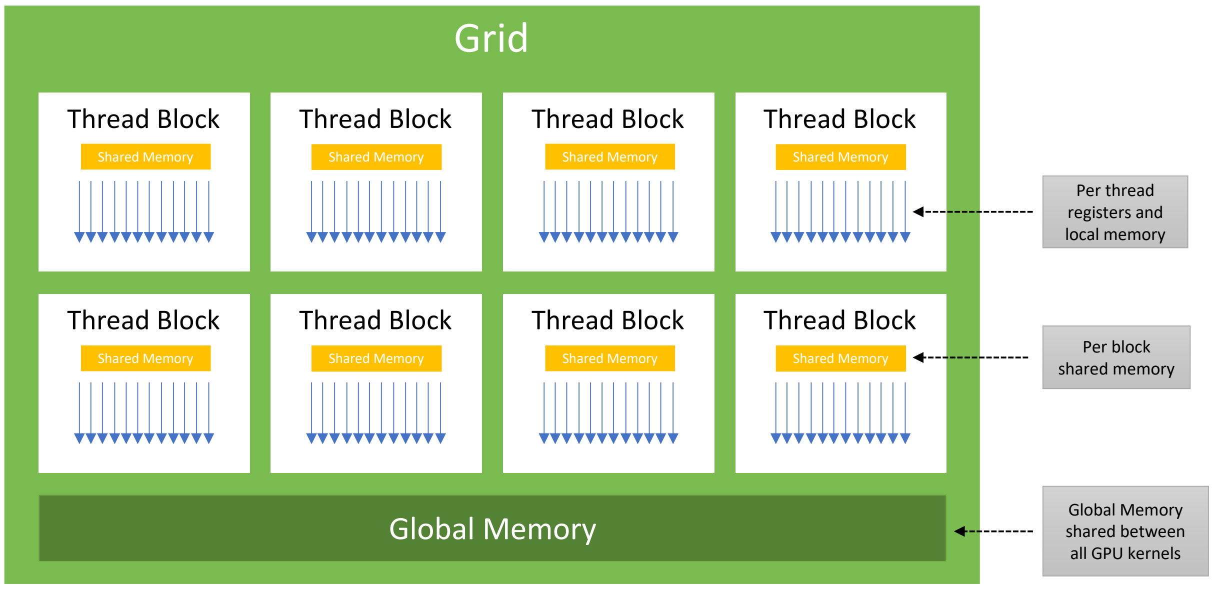

Grid
Thread Block
Shared Memory
Global Memory
Per thread registers and local memory
Per block shared memory
Global Memory shared between all GPU kernels

Figure 1.14: Thread Hierarchy and Memory Hierarchy [18, Chapter 2.2, 2.3]

In Subsection 1.7.2, we highlighted that CUDA's interpretation of memory hierarchy terminologies may differ when viewed from the computing framework's perspective compared to other contexts. Below is a concise breakdown of these terminologies [18, Chapter 5.3.2]:

- **Thread Register**: Serving as the fastest memory type, every thread maintains a private set of these registers. The available register space can influence the concurrent execution capability of threads.
- **Local Memory**: Each thread has its own dedicated local memory that isn't shared with others. For NVIDIA GPUs, it's important to note that despite its name, local memory is off-chip, making its access slower than shared memory or registers. Accesses to local memory are reserved for certain automatic variables.
- **Shared Memory**: On-chip and shared among threads within the same thread block, this memory type boasts faster access times than local or global memories.
- **Global Memory**: Positioned as the primary off-chip storage, it exhibits slower access times in contrast to on-chip memories. Nonetheless, it's universally accessible, permitting all threads to read and write.

To give readers a general insight into the utilization of CUDA and the execution of programs through this platform, we explore its collaboration with the CPU — an approach known as Heterogeneous Programming. Fundamentally, CUDA's programming model divides the hardware into two essential components: the host, which includes the CPU and its associated memory, and the device, representing the GPU and its dedicated memory.

Code running on the GPU is termed the "kernel" code, which is launched by the host and executed by the device. To illustrate this dynamic, we provide a C programming example that demonstrates the interplay between the CPU and GPU, as depicted in Figure 1.15. This illustration divides the computing system into two components: the Host and the Device. The Host initiates the Device, and while the Device is in execution, the Host can either continue its own tasks sequentially or employ specific commands, such as cudaDeviceSynchronize(), to pause the Host's thread and await the completion of the Device's execution. The key distinction here lies in the execution behavior: Host threads operate sequentially, while Device threads operate concurrently.

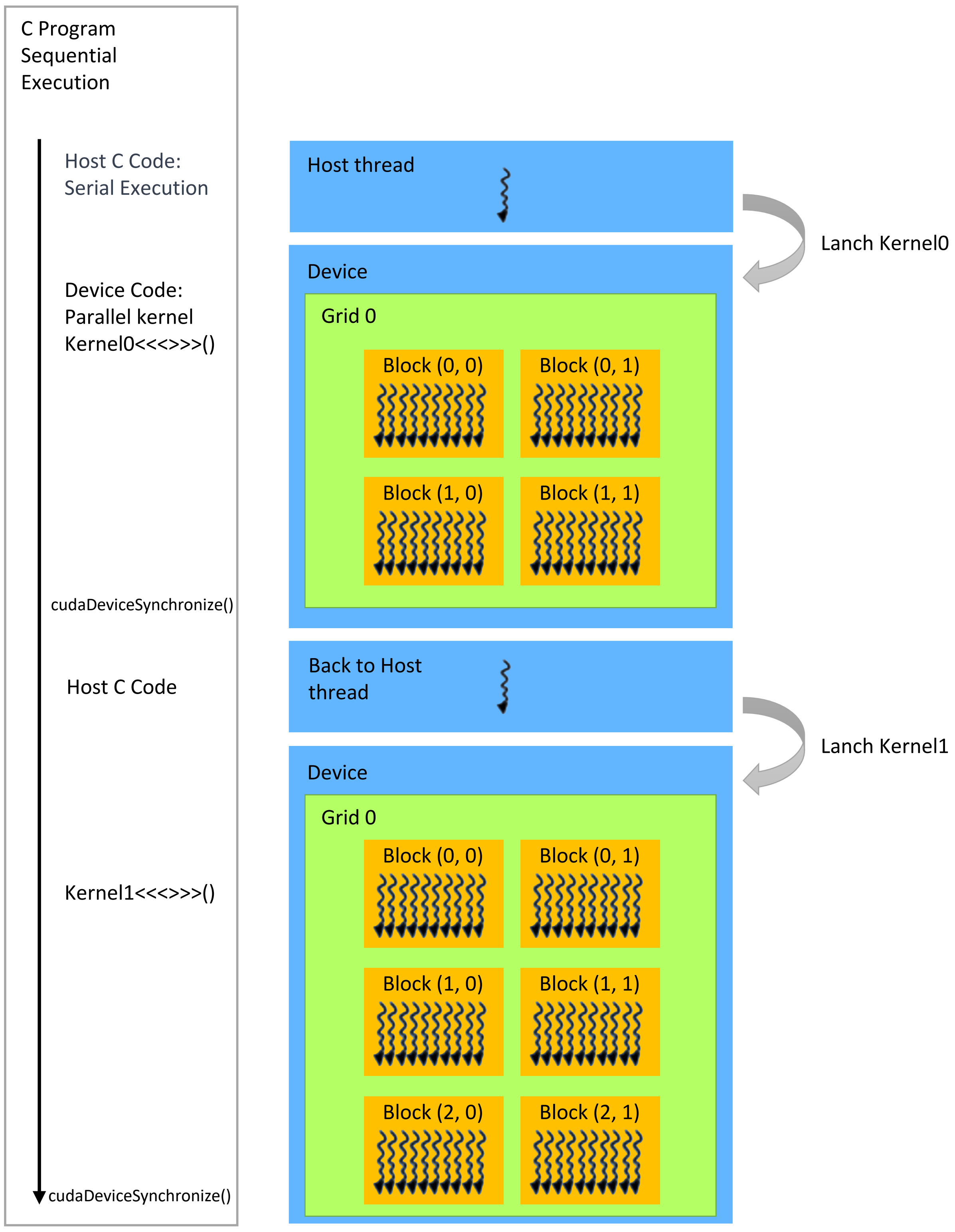


Figure 1.15: CUDA Heterogeneous Programming [18, Chapter 2.4]

1.7.4. Computing algorithm level

Within our High-Performance Computing level model, the Computing Algorithm level stands out as the most pivotal. If a program is designed with inherent inefficiencies, for instance, utilizing high-complexity algorithms, even the most optimal use of computer hardware or state-of-the-art large-scale infrastructure may not deliver desired outcomes. These inefficiencies not only compromise performance but also squander precious hardware resources. Consequently, we posit that the cornerstone of high-performance computing lies in a meticulously designed program. Hence, a best practice usually is to adopt a two-fold approach to crafting an effective program for your problem. First, address the computability aspect to ensure the problem can be solved algorithmically. Once this is confirmed, the subsequent step involves crafting a new algorithm or utilizing existing algorithms and data structures while aiming to minimize computational complexity.

Computability and complexity are foundational concepts in the theory of computation, each with its intricate and philosophical layers. Though closely related, complexity theory seeks to categorize problems as either 'easy' or 'hard', whereas computability theory differentiates between those that are solvable and those that are not [61, p.3]. In the current scenario, regarding computability, we presuppose the existence of an algorithm (or Turing machine) capable of solving every instance of the problem while directing our primary attention solely to the complexity aspect.

Within the realm of High-Performance Computing, addressing complexity is inevitable. Even if a problem is decidable and theoretically computable, it might remain impractical to solve if it demands excessive time or memory resources. Complexity primarily delves into the time, memory, and other resources crucial for solving computational problems [61, p.275]. Among these, time complexity and space complexity are of paramount importance. While this discussion predominantly centers on time complexity, the principles, and notations applied to space complexity mirror those of the former.

Evaluation of time complexity primarily hinges on two methods: **worst-case and average-case analysis**. The worst-case analysis examines the longest running time for inputs of a specific length. Conversely, the average-case analysis assesses the average running time across inputs of a given length. Given this backdrop, it's important to note that the precise running time of an algorithm can be an intricate expression, leading us to typically resort to estimation. Delving further, one prevalent method, known as **asymptotic analysis**, aims to estimate the algorithm's running time on large inputs. This approach hinges on focusing on the highest order term of its runtime expression, overlooking its coefficient and any lower order terms, primarily because the highest order term stands out as the predominant factor for large inputs [61, p.276].

In asymptotic analysis, specific notations are employed, notably **Big-O notation and Small-o notation**, to denote an algorithm's complexity. Big-O notation indicates that one function is asymptotically "no more than" another. Conversely, to convey that one function is asymptotically "less than" another, we employ the Small-o notation. This differentiation between Big-O and Small-o is akin to the distinction between $\leq$ (less than or equal to) and $<$ (strictly less than) [61, p.277, p.278].
Note that, as previously discussed and illustrated in Figure 1.8, each layer in our hierarchical model is interdependent. The algorithmic complexity of the program cannot be solely equated with its computational performance, as it is also closely tied to the computational framework employed and the hardware used.

2 | State of the art

After conformational sampling using the selected search algorithm in the molecular docking process, the next critical step is to evaluate these conformations using a scoring function. This mathematical model predicts the binding affinity between a ligand and its receptor, playing a crucial role in the molecular docking process by enabling the ranking of ligands based on their predicted binding affinities. Broadly, scoring functions can be categorized into four types: Physics-Based, Empirical, Knowledge-Based, and Machine Learning scoring functions. The classical scoring functions, which include the first three categories, are defined based on their feature items and typically employ linear regression techniques. In contrast, Machine Learning-based scoring functions incorporate nonlinear regression through advanced machine-learning approaches [42].
In this chapter, we provide a detailed overview of these four types of scoring functions and discuss state-of-the-art scoring functions relevant to our approach. Our methodology draws upon, refines, or is inspired by these established techniques, incorporating improvements where necessary. Various Scoring functions will also be further compared and analyzed in the experimental results section with our proposed approach.

2.1. Physics-Based Scoring Functions

Essentially, Physics-based scoring functions can be categorized into three types: force field, solvation models, and quantum mechanics methods. Each comes with its own set of strengths and limitations.

2.1.1. Force Field Scoring Functions

The force field scoring function is the most classic method. It calculates the binding energy E_{bind} by summing up the van der Waals interactions (E_{vdW}, often reflecting the non-bonded attraction or repulsion between atoms) and electrostatic interactions (E_{elec}, which capture the interactions between charged entities) between protein-ligand atom pairs, as represented in Equation 2.1. This method predominantly accounts for the enthalpic contribution to energy. While it offers computational efficiency, its omission of entropy and solvent effects means the force field-based SF can be notably suboptimal [42, 46].

$$E_{bind} = E_{vdW} + E_{elec} \tag{2.1}$$

To improve the force field-based SF, one can incorporate the torsion entropy of ligands and consider the solvation/desolvation effects (ΔG_{solv}). These effects can be represented by both explicit and implicit solvent models [42], as formulated in Equation 2.2.

$$E_{bind} = E_{vdW} + E_{elec} + \Delta G_{solv} \tag{2.2}$$

2.1.2. Scoring Function of DOCK

In this context, we highlight the scoring function of DOCK as a representative example of a force field that takes into account the solvent effect. As illustrated in Equation 2.3, it adopts its energy parameters from the Amber force fields [31, 46, 70, 71]. Within this equation, the terms $\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}$ signify van der Waals (VDW) interactions, while $\frac{q_i q_j}{\varepsilon(r_{ij})r_{ij}}$ denote electrostatic or Coulombic interactions. Here, r_{ij} represents the distance between protein atom i and ligand atom j; A_{ij} and B_{ij} are the VDW parameters; and q_i and q_j signify atomic charges. Notably, the solvent effect is implicitly captured using a simple distance-dependent dielectric constant, $\varepsilon(r_{ij})$, in the Coulombic term [31].

$$E = \sum_i \sum_j \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{\varepsilon(r_{ij})r_{ij}}\right) \tag{2.3}$$

While this example incorporates the solvent effect through a distance-dependent dielectric factor, it doesn't sufficiently account for the desolvation effect. According to the research by Huang et al. [31], the charged groups prefer aqueous environments, whereas non-polar groups gravitate towards non-aqueous settings. The desolvation energy, being a many-body interaction term, is contingent on the specific geometric and chemical contexts surrounding solute atoms. Overlooking the desolvation effect can skew a scoring function towards Coulombic electrostatic interactions, leading to a bias for highly charged ligands.

2.1.3. Quantum Mechanics (QM) Related Scoring Functions

Regardless, the aforementioned scoring functions are rooted in force fields. Their main limitation is that the predictive accuracy for binding energy heavily depends on the functional form of the potential energy and related parameters, which are challenging to pinpoint [42]. Hence, scoring functions (SF) rooted in quantum mechanics (QM) or the hybrid quantum mechanical/molecular mechanics (QM/MM) approach are formulated to address the intricacies of covalent interactions, polarization, and charge transfer in docking [42], as illustrated in Equation 2.4.

$$E_{bind} = E_{QM/MM} + \Delta G_{solv} \tag{2.4}$$

The quantum mechanics (QM) based SF offers a substantial improvement in prediction accuracy over the Force field SF. However, its heightened computational demand remains a significant challenge, even when employing the hybrid quantum mechanical/molecular mechanics (QM/MM) approach [15].

2.2. Empirical Scoring Functions

Empirical scoring functions (SFs) estimate protein-ligand binding affinity by accumulating energetic components such as hydrogen bonds, hydrophobic effects, and steric clashes. These functions are formulated as linear combinations of terms that depict different energy interactions at the protein-ligand interface, though specific terms can differ among scoring functions [10, 22, 25, 36, 69]. Empirical scoring functions have an evident drawback. The weights of the energetic components are determined by multivariate regression analysis using a training set with known binding affinities. Consequently, the accuracy of empirical SFs is profoundly anchored to the quality of the experimental data, which may introduce biases [36, 39]. Conversely, empirical scoring functions also present several notable advantages. Firstly, being calibrated with a diverse set of protein-ligand complexes enhances their adaptability beyond specific ligands or proteins. Secondly, each term not only has a clear physical meaning but, when combined with its assigned weight, our understanding of the receptor-ligand binding process is enhanced. Lastly, they are capable of achieving a prediction accuracy level of around 2 kcal/mol in binding affinity, which is deemed suitable for many structure-based virtual screening endeavors [13].

2.2.1. X-Score

A practical example of an empirical scoring function is X-Score, as shown in Equation 2.5. For simplicity, we've consolidated the hydrophobic term HP from three separate algorithms into a single term, HP. For a comprehensive presentation of X-Score, refer to

Wang et al. [69].

$$
\begin{aligned}
X - Score = w_0 \\
&+ w_{VDW} \cdot \text{VDW} \\
&+ w_{H-bond} \cdot \text{HB} \\
&+ w_{rotor} \cdot \text{RT} \\
&+ w_{hydrophobic} \cdot \text{HP}
\end{aligned}
\tag{2.5}
$$

In this equation 2.5, each w represents a weight determined during training process. The terms VDW, HB, RT, and HP correspond to the free energy change of van der Waals interaction, hydrogen bonding, deformation effect, and hydrophobic effect, respectively.

2.3. Knowledge-Based Scoring Functions

Knowledge-based scoring functions extract pairwise potentials from the three-dimensional configurations of numerous protein-ligand complexes, using the inverse Boltzmann statistical principle. The underlying assumption is that the frequency of various atom pairs at distinct distances correlates with their interaction, which is then translated into a distance-dependent potential of mean force [42]. Similar to the empirical scoring function, the knowledge-based scoring function encompasses both training and prediction processes. In the training phase, the function counts occurrences of each protein-ligand complex to construct the model. During prediction, it retrieves the pairwise atomic occurrences of the protein-ligand complex from the model and then follows the computing workflow to compute the binding affinity [28].

Knowledge-based SFs provide several advantages. Foremost, they effectively balance computational costs with predictive accuracy, distinguishing them from physics-based and empirical SFs. Additionally, the potentials derived from knowledge-based SFs are reliant solely on structural information, eschewing the need for experimental binding affinity data. This sidesteps potential ambiguities in binding affinities due to varying experimental conditions. Consequently, knowledge-based SFs are better suited for predicting binding poses rather than the binding affinities themselves [42].

However, knowledge-based SFs come with their own set of challenges. A pivotal component in these scoring functions is the 'reference state', which serves as a baseline or standard condition against which other states are compared. The determination of this reference state is not straightforward, primarily because it is an abstract representation that might not have a direct physical analog. Within the computational process, pinpointing an accurate reference state often proves elusive, given the myriad of factors that can influence molecular interactions. To circumnavigate this challenge, two predominant strategies have emerged. The first approximates the reference state by analyzing the random distribution of atomic pairs within the training set. In contrast, the second strategy refines this approximation using correction factors, such as the volume factor correction and physics-based iterative methods, to bolster accuracy [42].

In this section, DrugScore is highlighted as a specific example of a knowledge-based scoring function. Since its introduction in 2000, DrugScore has undergone significant refinements across three primary versions. Each of these versions was developed by the research group at Heinrich-Heine-Universität Düsseldorf, led by Prof. Dr. Holger Gohlke. Throughout its evolution, while the core principles of the original DrugScore remained intact, each iteration introduced unique enhancements and features, aiming to address evolving challenges in the field.

To illustrate the differences among the three versions, the original DrugScore classifies atoms into 17 types using the Sybyl atom types and incorporates both atom-pair potentials and Solvent-Accessible Surface (SAS) potentials. While they started with 6,026 protein-ligand complexes, the final training set might be fewer due to specific exclusionary rules applied [28]. On the other hand, DrugScoreCSD stands apart by deriving its pair-wise potentials from small-molecule crystal structures in the Cambridge Structural Database (CSD), as opposed to protein-ligand complexes. It includes 18 atom types, notably adding an atom type for iodine, and omits the use of SAS-potentials [67]. Building on these advancements, DrugScore2018 expands the model to encompass 25 atom types and utilizes an extensive training set of roughly 40,000 protein-ligand complexes from the Protein Data Bank (PDB). Importantly, similar to DrugScoreCSD, DrugScore2018 also does not consider SAS-potentials [21].

Further details of these versions will be discussed in the subsequent subsections.

2.3.1. DrugScore

DrugScore [28] utilizes training data from the ReLiBase database system, a three-dimensional platform focused on storing and analyzing protein-ligand complex structures [29]. This database houses 6,026 PDB protein structures, each containing ligands annotated with bond and atom types based on the SYBYL type convention. Gohlke et al. [28] applied a set of specific filtering rules to the dataset, each addressing a distinct purpose. For instance, to avoid the influence of atypical drug molecule sizes on the model, they excluded complexes with covalently bound ligands or those with ligands comprising fewer than six or more than 50 non-hydrogen atoms (heavy atoms). In another instance, given the first pass effect in the liver, they set a stringent upper limit of about 600 Daltons for a typical organic molecule. Additionally, they excluded any complexes later used to assess

predictive accuracy.
In the model, they assigned 17 Sybyl types to atoms, as outlined in Table 2.1. S.2 (sulfur sp2) was grouped with S.3, and N.4 (positively charged nitrogen) with N.3, mainly due to their rare presence or ambiguous assignment criteria. The training process primarily

Symbol	**Description**
C.3	Carbon sp3
C.2	Carbon sp2
C.ar	Carbon in aromatic rings
C.cat	Carbon in amidinium and guanidinium groups
N.3	Nitrogen sp3
N.ar	Nitrogen in aromatic rings
N.am	Nitrogen in amid bonds
N.pl3	Nitrogen in amidinium and guanidinium groups
O.3	Oxygen sp3
O.2	Oxygen sp2
O.co2	Oxygen in carboxylate groups
S.3	Sulfur sp3
P.3	Phosphorus sp3
F	Fluorine
Cl	Chlorine
Br	Bromine
Met	Ca, Zn, Ni, Fe

Table 2.1: 17 Sybyl atom types used in DrugScore.

involves counting occurrences for each protein-ligand atom type pair. Specifically, the occurrence $N_{i,j}(r)$ at each distance r is calculated from every Protein-Ligand complex, as shown in Equation 2.6.

$$N_{i,j}(r) = \sum_i \sum_j \delta\left(\left|\vec{r}_i - \vec{r}_j\right|, r\right) \tag{2.6}$$

In Equation 2.6, i and j represent individual atom types of the protein and ligand within the complex, respectively. The expression $|\vec{r}_i - \vec{r}_j|$ denotes the Euclidean distance between atom i and j. The function δ is a form of the Dirac delta function that assumes the value 1 when $r \leqslant |\vec{r}_i - \vec{r}_j| \leqslant r + dr$ and 0 otherwise. The bin size dr should be strategically chosen to ensure both a high resolution and a sufficient data sample within each bin. According to the findings of Gohlke et al. [28], a dr value of 0.1 Å was determined to produce satisfactory results.
In the model, the distance r is constrained between a minimum value r_{min} of 1 Å and a maximum r_{max} of 6 Å. The choice of 1 Å as the minimum accounts for typical metal-to-oxygen/nitrogen contacts, which typically occur around 1.8 Å. For distances shorter

than 1 Å, such proximate contacts will not be present in both crystal structures and computer-docked complexes due to van der Waals repulsion. The choice of 6 Å as the upper limit is motivated by the intention to focus on the geometrical discrimination of various ligand binding modes. This specific threshold ensures that highly specific interactions are dominant. Additionally, another advantage of selecting 6 Å is that it is short enough to exclude the possibility of a water molecule acting as a mutual mediator in a ligand-to-protein interaction.

It's important to acknowledge the inherent uncertainties in experimental data during the training process. As noted by Kossiakoff et al. in 1992 [41], atom position inaccuracies can reach up to 0.4 Å for a resolution of 2.5 Å. Given a bin size of 0.1 Å, such a distance becomes significant and cannot be overlooked. Gohlke et al. [28] employed a special smoothing function to address these uncertainties. They utilized a triangular weighting scheme to spread a single hit across multiple bins. Specifically, they allocated one hit to all adjacent bins within 0.2 Å, with the weight linearly decreasing from one to zero from the targeted bin.

Subsequently, the prediction phase involves computing the total score using the model, as presented in Equation 2.7. This equation primarily consists of two components: the atom-pair potential and the Solvent-Accessible Surface (SAS) potential. The atom-pair potential, represented as $\sum_{k_i}\sum_{l_j}\Delta W_{i,j}(r)$, is calculated as the sum of all pairwise potentials between protein and ligand atoms at a given distance r. The SAS potential, expressed as $\sum_{k_i}\Delta W_i\left(SAS, SAS_0\right)+\sum_{l_j}\Delta W_j\left(SAS, SAS_0\right)$, accounts for the sum of the SAS potentials for all individual protein and ligand atoms

$$\begin{aligned}\Delta W =&\gamma\sum_{k_i}\sum_{l_j}\Delta W_{i,j}(r)+\\ &(1-\gamma)\times\left[\sum_{k_i}\Delta W_i\left(SAS, SAS_0\right)+\sum_{l_j}\Delta W_j\left(SAS, SAS_0\right)\right]\end{aligned} \tag{2.7}$$

Within this context, the coefficient γ serves as a tunable parameter, which has been empirically optimized to a value of 0.5.

For the atom-pair potential term, it is reasonable to assume that the total preference, ΔW, for a specific binding geometry can be approximated by summing the individual contributions. This includes contributions from k_i ligand atoms of type i and l_j protein atoms of type j. Specifically, this relationship is expressed in Equation 2.8.

$$\Delta W_{i,j}(r) = W_{i,j}(r) - W(r) = -\ln\left[g_{i,j}(r)\right] - \left(-\ln\left[g(r)\right]\right) = -\ln\frac{g_{i,j}(r)}{g(r)} \tag{2.8}$$

Within this context, $\Delta W_{i,j}$ represents the net statistical preferences, derived from comparing the mean statistical preferences of subsystems, $W_{i,j}$, to the reference system W. Further, $W_{i,j}(r)$ denotes the absolutely pairwise potential between atom types i and j at a distance r. It corresponds to $g_{i,j}(r)$, which signifies the normalized radial pair distribution function for atoms of types i and j separated by a distance in the interval $[r, r + dr]$, as depicted in Equation 2.9. For $N_{i,j}(r)$, one has to compute the distance r between atoms i and j and then extract the count of occurrences from the model.

$$g_{i,j}(r) = \frac{N_{i,j}(r)/4\pi r^2}{\sum_r \left(N_{i,j}(r)/4\pi r^2\right)} \tag{2.9}$$

On the other hand, $W(r)$ serves as the reference system, encapsulating all non-specific information relevant to all atom pairs found in typical protein environments. Meanwhile, $g(r)$ is the normalized mean radial pair distribution function for a distance between any two atoms within the range $[r, r + dr]$, as illustrated in Equation 2.10.

$$g(r) = \frac{\sum_i \sum_j g_{i,j}(r)}{ij} \tag{2.10}$$

Additionally, both radial distribution functions (as per Equations 2.9 and 2.10) are normalized using the volume $4\pi r^2$ of the corresponding spherical shell for the interatomic distance r. This normalization facilitates quicker convergence to zero at greater distances [8].

In the original DrugScore, the Solvent-Accessible Surface (SAS) potential term is considered. This feature is not present in its subsequent versions, DrugScore CSD and DrugScore2018. As illustrated in Equation 2.7, both the Protein and Ligand atoms independently compute their respective SAS potentials, which are then aggregated. The underlying algorithm for computing the SAS potential of each atom is presented in Equation 2.11.

$$\Delta W_i\left(SAS, SAS_0\right) = W_i(SAS) - W_i\left(SAS_0\right) = -\ln \frac{g_i(SAS)}{g_i\left(SAS_0\right)} \tag{2.11}$$

For this equation, g_i represents the normalized distribution function of the surface area for atom i in its buried state (SAS), compared to its solvated state (SAS_0), considering both ligand and protein separately. Consequently, g_i is derived using an approximate cube algorithm, as depicted in Equation 2.12.

$$g_i(SAS) = \frac{N_i(SAS)}{\sum_{SAS} N_i(SAS)} \quad or \quad g_i\left(SAS_0\right) = \frac{N_i\left(SAS_0\right)}{\sum_{SAS_0} N_i\left(SAS_0\right)} \tag{2.12}$$

More specifically, $g_i(SAS)$ denotes the probability of observing an atom of type i with an exposed solvent-accessible surface (SAS) within a complexed state. In contrast, $g_i(SAS_0)$ conveys the probability of finding that same atom, having an equivalent solvent-accessible surface, in a fully dissociated state. In this context, $N_i(SAS)$ is the count of instances where atom i presents a solvent-accessible surface in the complexed state, whereas $N_i\ (SAS_0)$ tallies the instances of atom i revealing a solvent-accessible surface when completely unbound. The denominator acts as a normalization factor, accounting for the cumulative occurrences across both conditions.

2.3.2. DrugScore CSD

DrugScoreCSD [67], while sharing a similar formalism with the original DrugScore, distinguishes itself primarily in the derivation of its distance-dependent pair potentials. Unlike the original which sources these potentials from protein-ligand complexes, DrugScoreCSD derives them from nonbonded interactions observed in small organic molecule crystal packings. Opting for this approach offers notable benefits: the higher resolution of small molecule structures delivers more balanced distributions of atom types in contact data. Consequently, it yields potentials with enhanced statistical significance and finer shape details.
Regarding specifics, DrugScoreCSD utilizes data from the Cambridge Structural Database (CSD). The required entries are queried using the ConQuest engine. From the CSD, 28,642 entries were considered as crystal packings. For each entry, one central molecule is fully embedded within a comprehensive contact environment comprised of neighboring molecules. Each molecule within the crystallographic asymmetric unit was examined once to determine its nonbonded contact distances to all surrounding molecules in the crystal packing. For units with multiple molecular entries, molecules outside the asymmetric parts were included in the packing. As illustrated in Figure 2.1, the depicted calcium-acetylacetone complex features a calcium ion at the center, represented by the green ball. For deriving the pair potentials, only the acetylacetone and the calcium ion are considered as central molecules. In this framework, complexed metal ions are treated as independent "molecules". Uncomplexed water molecules, nonmetal ions, and molecules containing fewer than six heavy atoms are excluded.
For compatibility with the original DrugScore, DrugScoreCSD uses the same atom types for each atom in the crystal packing, as listed in Table 2.1. The sole exception is the introduction of a new atom type for iodine (I). As for its computation approach, DrugScoreCSD prioritizes the pairwise potential and omits the SAS potential. For the details of the pairwise potential procedure, one can refer directly to the original DrugScore's equations,

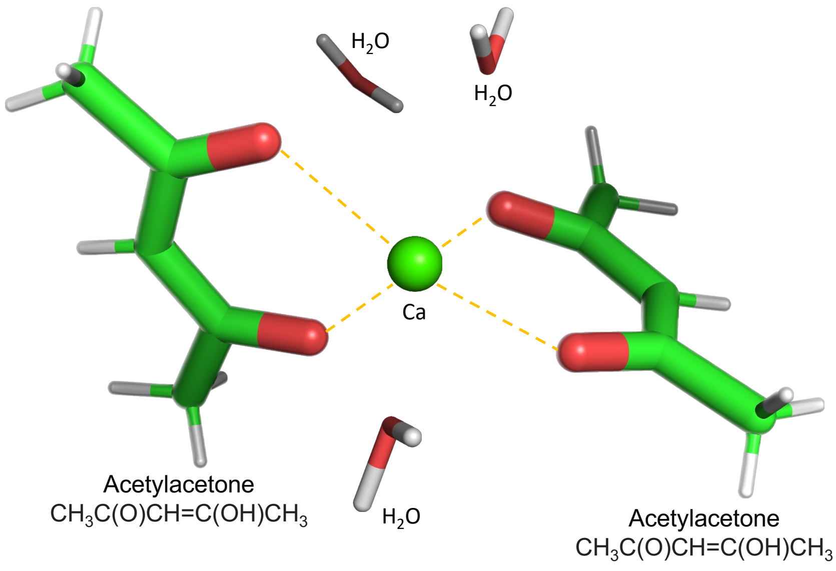


Figure 2.1: Example of a crystal packing used for the derivation of pair potentials, adapted from [67].

spanning from Equation 2.6 to 2.10.

2.3.3. DrugScore 2018

Developed by Dittrich et al., DrugScore2018 [21] is an enhanced version of its predecessor. Its most notable advancement is the expansion of the original 17 atom types in DrugScore to 25. The eight additional types introduced in this version are detailed in Table 2.2. Another enhancement in DrugScore2018 is its utilization of a more extensive training dataset, reflecting its expanded model capacity. The data is sourced from the Protein Data Bank (PDB), incorporating close to 40,000 protein-ligand complexes.

Atom Type	**Description**
I	Iodine
C.1	sp-hybridized Carbon (e.g., in alkynes)
N.1	sp-hybridized Nitrogen (e.g., in a nitrile group)
N.2	sp2-hybridized Nitrogen
N.4	Protonated sp3-hybridized nitrogen (e.g., protonated amino groups)
S.2	sp2-hybridized Sulphur
S.O	Sulphoxide sulphur
S.O2	Sulphone sulphur

Table 2.2: Newly introduced atom types in DrugScore2018.

In reality, the PDB contains much more than 40,000 protein-ligand complexes. The method DrugScore2018 employs to select suitable protein-ligand complexes and assess the quality of the selected dataset is notably worth mentioning. To identify the most suitable protein-ligand complexes for deriving the potentials, the following criteria were established:

- Retain only entries having a resolution of 2.5 Å or better, ensuring that structural features separated by at least 2.5 Å can be confidently discerned.
- Exclude complexes containing ligands that appear in the PDB more frequently than 500 times to mitigate potential biases in the derived potentials.
- Remove ligands with fewer than 10 or more than 100 heavy atoms. Additionally, ligands with unresolved (missing) atoms were excluded.
- Restrict the selection to no more than four identical ligands per PDB entry. It's important to note that while certain ligands may be excluded, they can still be present within the receptor structure.

On the other hand, DrugScore2018 evaluates the quality of the selected dataset by emphasizing two key aspects: ligand diversity and the impact of the predominant protein cluster on the potentials. To address ligand diversity, DrugScore2018 incorporates two evaluation criteria. First, the Tanimoto indices, as discussed in subsection 1.2.1, are utilized to measure the similarity among ligands. Second, combine the principal component analysis (PCA) and molecular quantum numbers (MQNs) [6] to gauge the chemical space coverage of the ligands. A high ligand similarity can lead to biases in the potentials derived, while a limited span in the chemical space can substantially hinder the model's capacity for generalization.

In the potential derivation process, DrugScore2018, much like DrugScoreCSD, focuses exclusively on the Pairwise Potential, omitting the SAS-Potential. This streamlined approach significantly enhances computing efficiency. Most of the derivation procedure aligns with the original DrugScore, as outlined from Equation 2.6 to 2.10. However, a minor adjustment in DrugScore2018, which applies to Equation 2.9, is the inclusion of a dr term in the normalized radial pair distribution, as presented in Equation 2.13. Typically, dr is set to 0.1 Å.

$$g_{i,j}(r) = \frac{N_{i,j}(r)/4\pi r^2 dr}{\sum_r \left(N_{i,j}(r)/4\pi r^2 dr\right)} \tag{2.13}$$

2.4. Machine Learning Scoring Functions

The scoring functions we have discussed so far are typically referred to as classic scoring functions. While these incorporate elements reminiscent of machine learning, they do not qualify as pure machine-learning scoring functions. For example, the process of determining weights during the training phase of empirical scoring functions bears similarities to supervised learning. Similarly, the methodology used in knowledge-based scoring functions, which involves extracting occurrences from each protein-ligand complex to construct the model, can be likened to unsupervised learning. However, genuine machine learning-based scoring functions significantly differ from these empirical and knowledge-based scoring functions, which only integrate certain aspects of machine learning. True machine-learning scoring functions explicitly employ machine-learning algorithms, such as support vector machines, random forests, or gradient boosting, to predict the effectiveness of decoys as potential ligands directly.
The general training procedure for machine learning scoring functions is illustrated in Figure 2.2. Mirroring conventional machine learning workflows, it primarily involves two stages: data processing and model training. Initially, data is extracted and preprocessed from specific databases and then split into training, validation, and test sets. Subsequently, the designated model is trained, and its predictive accuracy is evaluated.

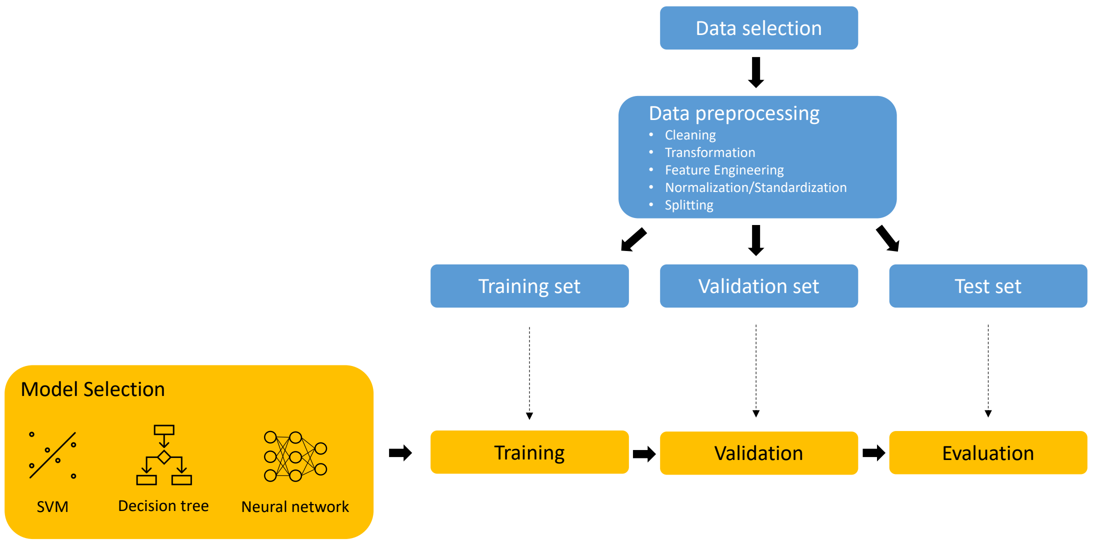


Figure 2.2: The general training process of machine learning scoring function.

While these trained models show promise in numerically evaluating binding affinity, their practical applications in drug discovery present certain limitations. In studies such as the one by Wang et al. [68], machine-learning based scoring functions excel at numerically

evaluating binding affinity compared to classical scoring functions. However, they don't consistently outperform in searching for correct ligands. As a result, they are rarely integrated directly into docking software but are typically employed for rescoring based on classic scoring functions [42, 74].
In machine learning, the primary tasks are classification and regression. However, within the realm of Drug Discovery, regression algorithms are used more frequently. They are usually employed to predict the binding affinity between target macromolecules (diseases) and ligand small molecules (drugs). Although various algorithms are available for regression tasks, our methodology predominantly incorporates decision tree-based algorithms. In this section, we delve deeply into the regression aspect of the two decision tree algorithms. Given that the Machine Learning Scoring Function is a direct application of Machine Learning algorithms in the Scoring Function domain, our discussion will primarily concentrate on the algorithms themselves.

2.4.1. Gradient boosting

Gradient Boosting, as described by Jerome H. Friedman in [24], is a robust machine-learning algorithm. Friedman developed it specifically for regression gradient boosting in 1999. The term 'Gradient' denotes its use of gradient descent to minimize loss, while 'Boosting' implies the method of sequentially training a series of weak learners. Each new learner specifically addresses the errors made by its predecessors. A weak learner is defined as a model that performs marginally better than random chance. In Gradient Boosting, decision trees, typically of shallow depth, are frequently employed as these weak learners.
Let's delve into the detailed methodology behind Gradient Boosting. Firstly, we denote our input data as $\{(x_i, y_i)\}_{i=1}^n$, where x_i represents the features and y_i signifies the labels. Typically, each instance might have multiple features but only one label. In this notation, the subscript i indicates the i-th sample, and the range $i = 1$ to n encompasses all samples, with n being the total number of samples. Following that, we must also define a differentiable loss function $L(y_i, F(x))$, where $F(x)$ represents the predicted value from the model. Commonly, the squared-error $(y - F(x))^2$ is used as the loss function. To simplify its derivative by canceling the square term, the loss function can be multiplied by $\frac{1}{2}$. This results in $\frac{1}{2}(y - F(x))^2$.
The main part of the Gradient Boosting is presented in Algorithm 2.1. This process will be repeated over M iterations, where M is a hyperparameter that requires careful selection. Too small a value for M can lead to underfitting, while an excessively large M risks overfitting. Such overfitting can yield high accuracy on the training set but compromises

the model's generalization capability.

Algorithm 2.1 Gradient Boosting by Friedman [24]

```
Require: Training data $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y_i, F(x))$, number
    of iterations $M$
1: Initialize model with a constant value: $F_0(x) = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, \rho)$
2: for $m = 1$ to $M$ do
3:     Compute "pseudo-responses": $\tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, i = 1, N$
4:     Fit a weak learner: $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} \left[\tilde{y}_i - \beta h\left(\mathbf{x}_i; \mathbf{a}\right)\right]^2$
5:     Find the best gradient descent step-size $\rho_m$:
       $\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L\left(y_i, F_{m-1}\left(\mathbf{x}_i\right) + \rho h\left(\mathbf{x}_i; \mathbf{a}_m\right)\right)$
6:     Update model: $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h\left(\mathbf{x}; \mathbf{a}_m\right)$
7: end for
Ensure: Model $F_M(x)$
```

Initially, the model is set to a constant value, which is determined by minimizing the loss function. For the squared-error, this optimal constant is simply the mean of all labels. Following this initialization, the algorithm begins its iterative process. It successively trains a series of weak learners, with each one focused on rectifying the errors from the preceding learners. At the start of each iteration, the algorithm calculates the pseudo-responses, denoted by $\tilde{y}_i$, for each sample. This computation is based on the derivative of the loss function with respect to the predicted value, $F(x)$, from the model of the previous iteration, $F_{m-1}(x)$. Given the influence of the loss function in this gradient boosting process, as opposed to linear regression, each $\tilde{y}_i$ should not be viewed as a normal residual. Instead, it is best understood as a 'pseudo residual'.
Subsequently, as outlined in the fourth step of Algorithm 2.1, we fit a new weak learner represented by the function $h\left(\mathbf{x}; \mathbf{a}\right)$. This function is a parameterized interpretation of the input variables x, defined by parameters $\mathbf{a} = \{a_1, a_2, ...\}$. The parameter $\mathbf{a}_m$ denotes the output value of the weak learner, with the subscript m indicating its index. The process of fitting varies based on the type of weak learner used. For example, when using a decision tree as the weak learner, the fitting process involves finding the optimal splitting variables and split locations. Similarly, the output values $\mathbf{a}_m$ also depend on the tree's splitting variables, split locations and means of the terminal nodes. However, the general optimization strategy is outlined in the equation presented in the fourth step. Within this equation, Friedman also introduces β as the scaling factor for the weak learner.
In the fifth step, we need to establish the optimal stepsize ρ, for gradient descent. This value is determined by minimizing the loss function, which evaluates the difference between the training data y_i and the combined output of the old model $F_{m-1}\left(\mathbf{x}_i\right)$ with a new weak learner $h\left(\mathbf{x}_i; \mathbf{a}_m\right)$ scaled by its stepsize ρ, as illustrated in Algorithm 2.1, Step

5. This step embodies the core principle of Gradient Boosting - operating as a greedy algorithm with a stagewise strategy, each new learner directly rectifies the errors of its predecessors. Notably, there is an essential difference between the stagewise approach used by Gradient Boosting and the stepwise approaches. The stagewise method incrementally adds to the model without adjusting previous inclusions. In contrast, the stepwise approach might adjust or even omit earlier terms based on specific criteria when introducing new components to the model.
Finally, in the last step, the model is updated by incorporating the new weak learner $\rho_m h\left(\mathbf{x}; \mathbf{a}_m\right)$ into the previous model $F_{m-1}(\mathbf{x})$, as depicted in Algorithm 2.1, Step 6. This process continues until the desired number of iterations is reached. Notably, in practice, we often introduce the learning rate ν in this step, as highlighted in Equation 2.14.

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu\rho_m h\left(\mathbf{x}; \mathbf{a}_m\right) \tag{2.14}$$

This hyperparameter determines the magnitude of parameter updates during model training. A lower learning rate diminishes the influence of each weak learner on the overall model, potentially enhancing its accuracy. However, this can slow down the learning process, so striking the right balance is crucial.

2.4.2. Extreme Gradient Boosting (XGBoost)

Compared to the Gradient Boosting discussed in Subsection 2.4.1, Extreme Gradient Boosting (XGBoost) [16] introduces an "Extreme" dimension. It builds upon the foundational Gradient Boosting by incorporating features like regularization, parallel learning, and hardware optimization, elevating it from a basic algorithm to an industrialized tool. Consequently, industries can directly utilize XGBoost through the API developed by Chen et al. [16], benefiting from its high prediction accuracy and exceptional computing performance without navigating complex technical details. In this section, we will discuss XGBoost from two perspectives: first, the construction of its unique regression tree, and second, its optimization for hardware utilization.
Distinguishing itself from other Gradient Boosting algorithms, XGBoost employs a unique regression tree as its weak learner for regression tasks. This tree is tailored, integrating a regularization term to reduce overfitting and addressing issues related to expansive datasets, numerous features, and data sparsity. For optimal split finding, XGBoost provides three algorithms: the Exact Greedy Algorithm, the Approximate Algorithm, and the Sparsity-aware Split Finding algorithm. The Exact Greedy Algorithm evaluates all possible splits across every feature, making it accurate but less efficient for vast datasets.

Conversely, the Approximate Algorithm segments the data and uses these partitions specifically for split finding. This approach is more commonly used, especially considering the size of datasets in modern regression tasks. Finally, to further accommodate the challenges posed by missing values, leading to sparse datasets, XGBoost offers the Sparsity-aware Split Finding algorithm.

Algorithm 2.2 Exact Greedy Algorithm for Split Finding [16]

```
Require: $I$, sorted instance set of current node
1: $G_L \leftarrow 0$, $H_L \leftarrow 0$
2: for $j$ in range(1, len($I$)+1) do
3:     $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
4:     $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
5:     $score \leftarrow \max\left(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$
6: end for
7: Split with max score
Ensure: Updated Regression Tree
```

As outlined in Algorithm 2.2, the Exact Greedy Algorithm meticulously evaluates the split position for each node. Given I as the instance set of a node, it is sorted based on its feature attributes; G_L represents the sum of the residual values of instances in the left leaf and g_j represents the sum of the residual values of the newly added elements, while H_L and h_j indicate their counts. The residual value of an instance is the difference between its predicted value from the previous tree and its actual value. During the loop, splits are evaluated in sequence using the equation $\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}$ to determine the Gain for each potential split. Ultimately, the split with the highest gain is selected. After this decision, child nodes are re-evaluated. This iterative process continues until no viable splits remain or a predetermined number of iterations is reached. An integral aspect of this procedure is the incorporation of a regularization term λ during Gain calculation. In this context, term $\frac{G_L^2}{H_L+\lambda}$ is referred to as 'similarity'. A higher similarity indicates a more favorable split. The difference between the sum of the child node similarities and the parent node similarity defines the gain. By introducing the regularization term, the similarity is reduced, making the splitting process more conservative. This, in turn, helps manage the complexity of the regression tree. Be aware that Algorithm 2.2 doesn't cover all methods to manage the complexity of the regression tree, like tree pruning. Through the integration of the parameter 'gamma', once the tree is fully generated, any splits yielding a gain less than gamma are pruned away. In this way, combined with the regularization term, the complexity of the regression tree is further controlled.

Algorithm 2.3 outlines the Approximate Algorithm's process, which unfolds in two distinct stages. The first stage, encompassing Steps 1 to 4, identifies split points. The second

Algorithm 2.3 Approximate Algorithm [16]

```
1: for $k = 1$ to $m$ do
2:     Propose $S_k = \{s_{k1}, s_{k2}, ..., s_{kl}\}$ by percentiles on feature k.
3:     Proposal can be done per tree (global), or per split (local).
4: end for
5: for $k = 1$ to $m$ do
6:     $G_{kv} \leftarrow= \sum_{j\in\{j|s_{k,v}\geq x_{jk}\geq s_{k,v-1}\}} g_j$
7:     $H_{kv} \leftarrow= \sum_{j\in\{j|s_{k,v}\geq x_{jk}\geq s_{k,v-1}\}} h_j$
8: end for
9: Follow same step as in previous section to find max score only among proposed splits.
```

stage, involving Steps 5 to 8, focuses on mapping data points to their corresponding buckets and aggregating them. Specifically, the process starts by identifying candidate split points s_{kv}, where k denotes the feature index and v represents the index of split points, derived from the feature distribution percentiles. This is often achieved using the Weighted Quantile Sketch method, which typically ensures an even distribution of data across each quantile. Additionally, this step can be executed either globally or locally, with the distinction based on the timing of the proposals. The global method suggests all candidate splits initially and uses them throughout tree construction. In contrast, the local method updates its proposals after each split. While the global approach has fewer proposal steps, it typically needs more candidate points since they aren't refined post-split. The local method, which refines candidates post-split, might be better suited for deeper trees. After identifying the split points, the algorithm maps continuous features into distinct buckets, as outlined in Steps 6 and 7 of Algorithm 2.3. The expression $s_{k,v} \geq x_{jk} \geq s_{k,v-1}$ indicates that all data between $s_{k,v}$ and $s_{k,v-1}$ will be mapped into the same bucket. Each bucket is treated as an indivisible unit. Finally, the Approximate algorithm uses the buckets divided in this step to perform subsequent operations identical to those in Algorithm 2.2, instead of using scattered individual data points.

Next, Algorithm 2.4 outlines the Sparsity-aware Split Finding algorithm. The primary distinction between this algorithm and the Basic Exact Greedy Algorithm is its dual calculation of Gain during node splitting. Initially, it assigns all missing values to the right using $G_R \leftarrow G - G_L$; subsequently, it places them on the left using $G_L \leftarrow G - G_R$. Ultimately, the algorithm adopts the split associated with the highest Gain. This approach offers an ingenious and straightforward solution to handle missing values, significantly enhancing XGBoost's proficiency in addressing real-world challenges.

Furthermore, XGBoost employs three distinct strategies to optimize hardware utilization. The first strategy employed by XGBoost is the use of Column Block for Parallel Learning. In this approach, data is organized into in-memory units, termed as 'blocks'. Within

Algorithm 2.4 Sparsity-aware Split Finding [16]

```
Require: $I$, sorted instance set of current node
Require: $I_k = \{i \in I | x_{ik} \neq missing\}$
 1: // enumerate missing value goto right
 2: $G_L \leftarrow 0$, $H_L \leftarrow 0$
 3: for $j$ in range(1, len($I_k$)+1) do
 4:     $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
 5:     $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
 6:     $score \leftarrow \max\left(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$
 7: end for
 8: // enumerate missing value goto left
 9: $G_R \leftarrow 0$, $H_R \leftarrow 0$
10: for $j$ in range(1, range(len($I_k$), 0, -1) do
11:     $G_R \leftarrow G_R + g_j$, $H_R \leftarrow H_R + h_j$
12:     $G_L \leftarrow G - G_R$, $H_L \leftarrow H - H_R$
13:     $score \leftarrow \max\left(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$
14: end for
15: Split and default directions with max gain
Ensure: Updated Regression Tree
```

these blocks, data is stored column-wise using the Compressed Sparse Column (CSC) format, with each column sorted according to its respective feature value. This columnar structure facilitates parallel processing, enabling individual columns to be processed in parallel across multiple threads or within distributed systems.

Subsequently, XGBoost employs Cache-aware Access. As illustrated in Figure 2.3, modern computers utilize a hierarchical memory layout. The closer the memory is to the CPU, the faster it's accessed, albeit with a smaller capacity. When the CPU requires data, it first attempts the cache. A cache miss necessitates a more time-consuming query to the Main Memory. To address this, XGBoost strategically positions and prefetches frequently used data, such as gradient statistics, in cache memory nearer to the CPU. This approach reduces cache miss rates and thereby enhances performance.

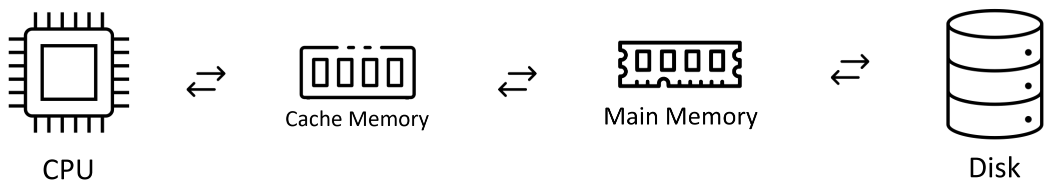


Figure 2.3: Memory Layout

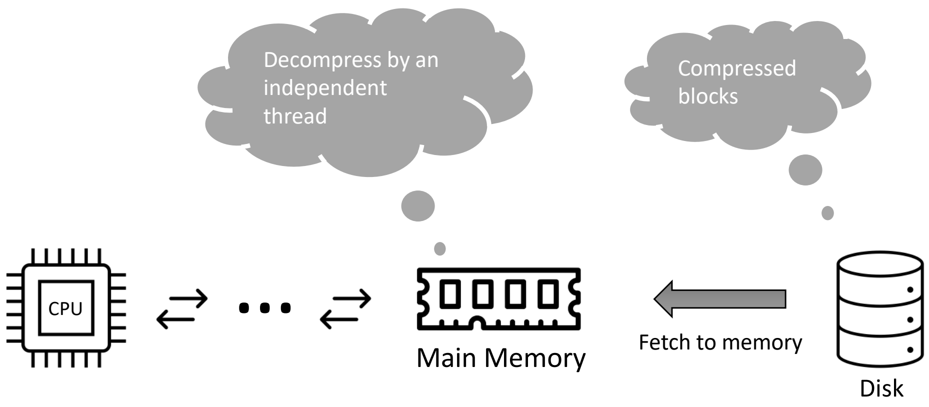
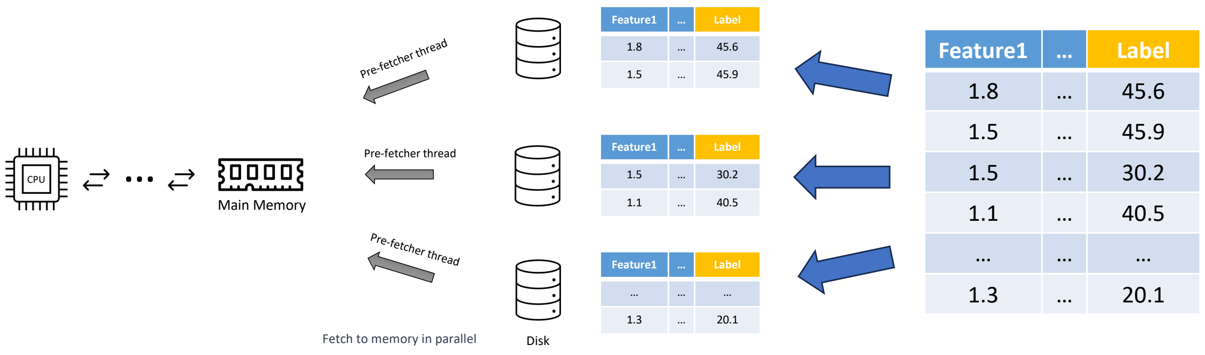


Figure 2.4: Block Compression

Lastly, XGBoost leverages Out-of-Core Computing, which is beneficial when handling datasets that exceed memory capacity. During computation, XGBoost employs a dedicated thread to prefetch blocks from the disk into the main memory buffer, allowing processing to occur in parallel with disk reading. Additionally, XGBoost incorporates Block Compression and Block Sharding techniques to further enhance its Out-of-Core Computing capabilities. Figure 2.4 illustrates the Block Compression technique. Blocks stored on the disk are compressed, and once fetched into the main memory buffer, an independent thread decompresses them. This strategic approach significantly reduces data transfer time, ensuring efficient data retrieval. On the other hand, the Block Sharding technique, depicted in Figure 2.5, partitions the data across multiple disks. Each disk is assigned a pre-fetcher thread to transfer the data into an in-memory buffer. Afterward, the training thread alternates reading from these buffers. When multiple disks are in use, this strategy notably enhances the system's efficiency, delivering a substantial boost to disk reading throughput.



Feature1	...	Label
1.8	...	45.6
1.5	...	45.9

Feature1	...	Label
1.5	...	30.2
1.1	...	40.5

Feature1	...	Label
...	...	...
1.3	...	20.1

Feature1	...	Label
1.8	...	45.6
1.5	...	45.9
1.5	...	30.2
1.1	...	40.5
...	...	...
1.3	...	20.1

Figure 2.5: Block Sharding

3 | Methodology

In this chapter, we will present our proposed methodology in detail. Our objective is to identify a set of optimal ligands for the target protein from a vast library of drug molecules and then forward them for subsequent in-vitro experimentation. A significant challenge in this multidisciplinary domain of Drug Discovery and High-Performance Computing (HPC) is striking a balance between prediction accuracy and computational performance. To address this challenge, our methodology took into account various HPC levels. At the computing algorithm level, we introduced DrugXGBScore, a hybrid scoring function that combines knowledge-based and machine learning approaches. This scoring function leverages the straightforward structure of the knowledge-based scoring function while benefiting from the potential predictive accuracy of machine learning scoring functions. On the other hand, at the Computing framework and hardware level, we employed CUDA to effectively deploy DrugXGBScore on our state-of-the-art HPC hardware.
This chapter is structured into two main parts. From Section 3.1 to Section 3.4, we provide a detailed exploration of our proposed hybrid scoring function, DrugXGBScore. In Section 3.5, we outline the deployment of DrugXGBScore within our HPC Pipeline.

3.1. Overview of the Hybrid scoring function: DrugXGBScore

Our proposed scoring function, DrugXGBScore, integrates two different types of scoring equations as a hybrid solution. This approach draws inspiration from the work of Wang et al. [68] and Lu et al. [43]. They integrated various types of scoring functions, specifically the empirical and machine learning scoring functions, to enhance prediction accuracy. However, a key distinction in our methodology is our imperative to address the HPC challenge, balancing prediction accuracy with computational performance, instead of merely prioritizing accuracy. To achieve this balance, we chose to combine the Knowledge-based and Machine learning scoring functions. The simpler structure of the Knowledge-based scoring function readily meets our HPC demands, while a well-trained Machine learning scoring function can further enhance prediction accuracy.

The general workflow of our hybrid scoring function, DrugXGBScore, is shown in Figure 3.1. The primary objective of this workflow is to score the Protein-Ligand Complex. Initially, the three-dimensional coordinates of all atoms in the Protein and Ligand, accompanied by their SYBYL atom types [21, 51], are inputted into the system. The subsequent calculation steps involve scoring with our optimized DrugScore2018 and XGBoost, and then combining these two scores to yield the final score. This final score quantifies the binding affinity of the Ligand to its Protein at that specific pose or position. A higher final score indicates greater binding affinity.

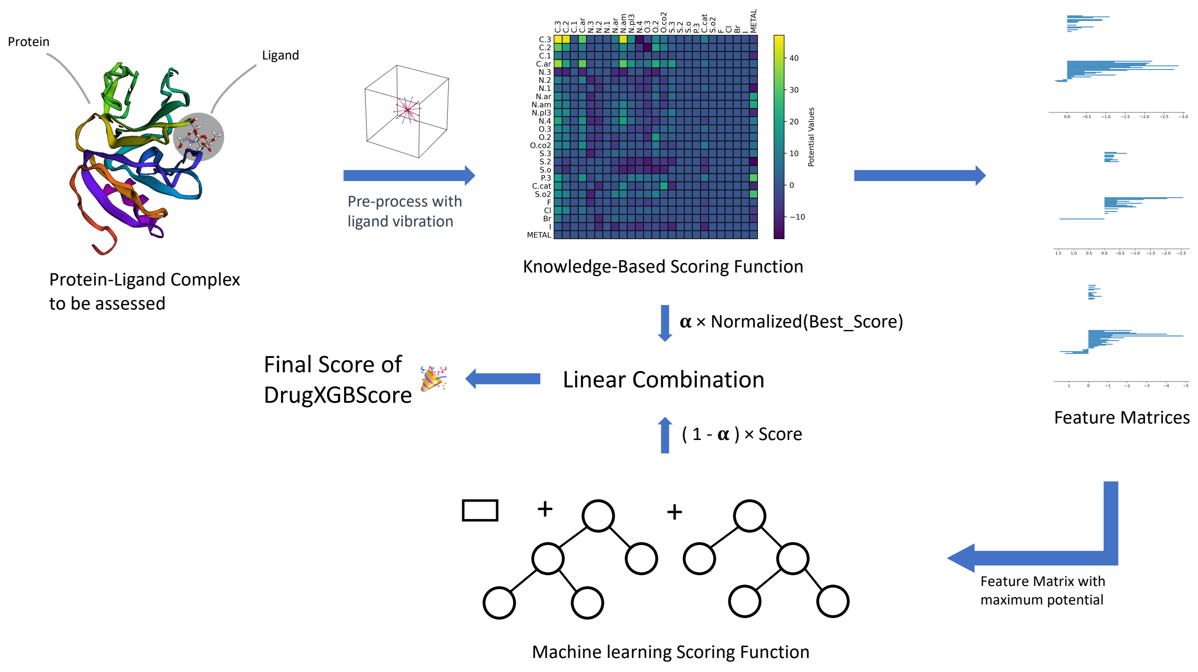

Protein
Ligand
Protein-Ligand Complex to be assessed
Pre-process with ligand vibration
Knowledge-Based Scoring Function
Potential Values
Feature Matrices
α × Normalized(Best_Score)
Final Score of DrugXGBScore
Linear Combination
(1 - α) × Score
Feature Matrix with maximum potential
Machine learning Scoring Function

Figure 3.1: The general workflow of DrugXGBScore

More specifically, for the Protein-Ligand complex under assessment, it is first scored using the Knowledge-based scoring function, DrugScore2018, which we have optimized as detailed in Section 3.2. In this step, we introduced a novel concept named 'ligand vibration'. By setting a specific radius, we maneuver the ligand in various directions within the complex to generate new conformations. These conformations are individually assessed, with the most optimal potential selected as the final result. This procedure can be likened to a simplified Molecular Dynamics (MD) simulation, mitigating inherent uncertainties from experimental data [41] and enhancing training set quality via these straightforward maneuvers. Subsequently, feature matrices are derived from Optimized DrugScore2018, and then the feature matrix with the highest potential is selected as input for the Machine Learning scoring function. For this task, we chose XGBoost as the Machine Learning algorithm. This is a renowned and extensively utilized approach, as elaborated in Section

2.4.2. Our choice of XGBoost is driven not only by its robust predictive prowess but also by its built-in high-performance computing capabilities. Finally, after normalizing the Knowledge-based and Machine Learning scores to the same scale, we employed a linear combination, characterized by the parameter α with the complementary weight being $1-\alpha$ to integrate them. Drawing inspiration from recommendation system algorithms, such linear combinations are prevalent methods for combining different recommender system approaches.

3.2. Optimization of Knowledge-based DrugScore2018

3.2.1. Data Selection and Preprocessing

We sourced the training set for this optimized version of DrugScore2018 from PDBBind v2020 [53]. This version comprises a total of 19,443 Protein-Ligand Complexes, with 5,316 in the "refined set" and 14,127 in the "general set". The data selection procedure comprises two steps. Initially, we implemented a preliminary data filtering, with certain aspects drawing from the methodology of Dittrich et al. [21]. The filtering strategies are detailed below:

- Excluded complexes with resolution > 2.5 Å.
- Excluded ligands with < 10 or > 100 heavy atoms (i.e., atoms other than Hydrogen).
- Excluded complexes in the PDBBind coreset (note: the coreset is used as the CASF-2016 test set).
- Excluded ligands that have a Tanimoto Similarity value > 0.2 and, concurrently, a frequency > 200.
- Excluded ligands with binding affinity values where -logKd/Ki < 6.0.

Two points from the list warrant further emphasis. First, to ensure ligand diversity, we employ Extended-connectivity fingerprints (ECFPs), as detailed in Section 1.4. Each ligand is assigned a fingerprint, and its Tanimoto Similarity value is then calculated against others. Ligands exhibiting a Tanimoto Similarity value greater than 0.2 are ranked by their frequency of occurrence. A higher frequency suggests that the ligand frequently exhibits high similarity with other ligands. If a ligand appears more than 200 times in this list, it is excluded. Second, given that the Knowledge-based scoring function operates similarly to an Unsupervised learning process, we aim to mitigate the undue influence of low binding affinity ligands. Thus, ligands with binding affinity values of -logKd/Ki less than 0.6 are excluded.

After applying the filtering strategy outlined above, the training dataset was narrowed down to 6,048 complexes. Subsequently, we conducted control variable experiments on these complexes. Under consistent conditions, we trained models on different sizes of training subsets and assessed their predictive capabilities. The final training set was chosen based on the model exhibiting the top prediction accuracy. It's important to mention that, according to the PDBBind developers [53], the refined set typically holds a higher quality compared to the other set. Given this, our selection process gave precedence to the inclusion of complexes from the refined set. In the end, the final training set is composed of 2,900 complexes: 1,146 from the 'refined set' and 1,754 from the 'general set'. The results of the control variable experiments can be found in Section 4.1.
After completing the data selection, all the chosen protein-ligand complexes need to be preprocessed. A key component of this step is the accurate assignment of the SYBYL atom type [21, 51] to every atom in both the protein and ligand. The precision of this assignment directly impacts the dataset's quality. For this purpose, we evaluated various tools, including OpenBabel [64] and Fconv [49]. Due to potential ambiguities inherent in the SYBYL atom type, the assignment results were not always satisfactory. After assessing multiple tools, we settled on OpenBabel. The finalized atom types that we used in our optimized version of DrugScore2018 are detailed in Table 3.1.

3.2.2. Training process

The training process for our optimized version of DrugScore2018, as depicted in Figure 3.2, primarily comprises two phases: calculating "Occurrences" and "Potentials". In the Occurrences calculation phase, we opted to process each Protein-Ligand complex individually before consolidating the results. This method offers more flexibility than the traditional approach of accumulating complexes sequentially. It enables the easy selection of training subsets, inclusion or exclusion of specific complexes, and expedites the subsequent model evaluation process. The calculation of Occurrences adheres to the original DrugScore method, as detailed in Section 2.3.1, Equation 2.6. In this process, we calculate the Euclidean distance between each protein and ligand atom. If the distance between a specific protein and ligand atom pair falls within the threshold range of 1.0 to 6.0, it's considered a hit, indicating an "Occurrence" at that distance. Regrettably, as discussed in Section 2.3.1, the interaction 'hit' between the protein and ligand carries inherent uncertainty. As noted by Kossiakoff et al. [41], for a protein-ligand complexes resolution of 2.5 Å, inaccuracies in atom positions can be as substantial as 0.4 Å. To address this inherent uncertainty, Gohlke et al. [28] introduced a smoothing function known as the Triangular Weighting Scheme, illustrated in Figure 3.3 A. In this approach, a single

Symbol	Description
C.1	sp-hybridized Carbon
C.2	Carbon sp2
C.3	Carbon sp3
C.ar	Carbon in aromatic rings
C.cat	Carbon in amidinium/guanidinium groups
N.1	sp-hybridized Nitrogen
N.2	Nitrogen sp2
N.3	Nitrogen sp3
N.4	Protonated Nitrogen sp3
N.ar	Nitrogen in aromatic rings
N.am	Nitrogen in amide bonds
N.pl3	Nitrogen in amidinium/guanidinium groups
O.2	Oxygen sp2
O.3	Oxygen sp3
O.co2	Oxygen in carboxylate groups
S.2	Sulphur sp2
S.3	Sulphur sp3
S.o	Sulphoxide sulphur
S.o2	Sulphone sulphur
P.3	Phosphorus sp3
F	Fluorine
Cl	Chlorine
Br	Bromine
I	Iodine
Met	All built-in metal atom types in OpenBabel

Table 3.1: The 25 SYBYL atom types [21, 51] utilized in our optimized DrugScore2018.

'hit' is distributed across several bins. Specifically, a hit is assigned to all neighboring bins within a 0.2 Å radius of the central bin α, with its weight linearly diminishing from one to zero across that range [28]. Additionally, in our investigations for the optimized version of DrugScore2018, as depicted in Figure 3.3 B, we also experimented with a more direct approach called the Full Weighting Scheme. In this approach, a single hit is uniformly distributed to all adjacent bins. For a detailed evaluation of their impacts, please refer to Section 4.1.

After calculating the occurrences for each complex, we integrate them to derive the Potentials. The resulting potential values for each atom pair at various distances constitute our final model. Similar to the occurrences calculation, the potential calculation also follows the original DrugScore method in Section 2.3.1, as illustrated in Equations 2.8

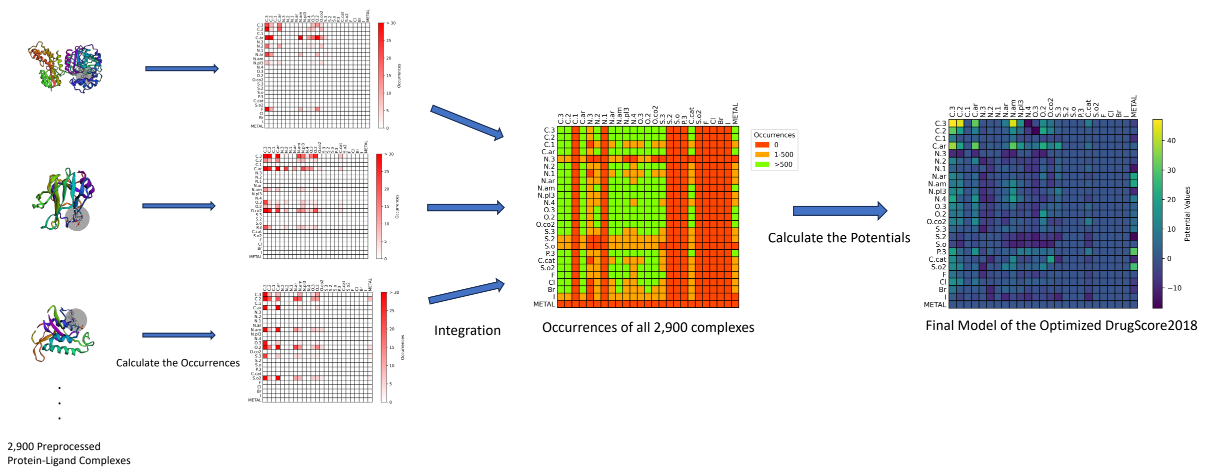
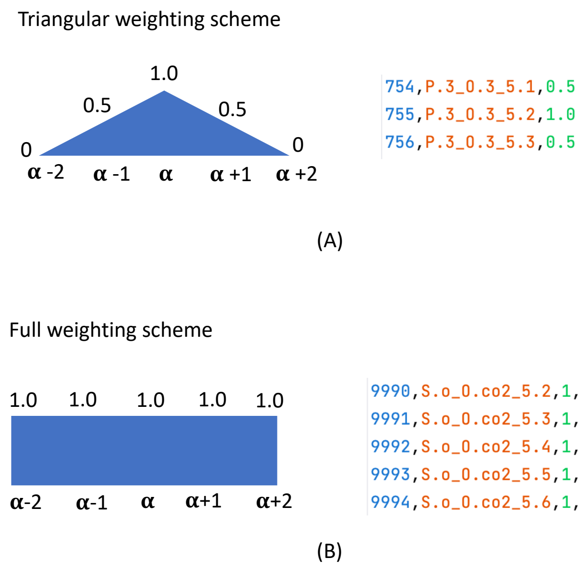


Figure 3.2: Training process of Optimized DrugScore2018

to 2.10. It's worth noting that, in contrast to the original DrugScore, we excluded the Solvent-Accessible Surface (SAS) potential term and focus solely on the pairwise potentials between protein and ligand atoms.

3.2.3. Inference process

The inference process of our optimized DrugScore2018 is illustrated in Figure 3.4. The input for this process consists of the 3D coordinates of all atoms, along with their SYBYL atom types [21, 51], in the Protein-Ligand complex, and the output is a score and a feature matrix, which will be utilized in the subsequent XGBoost phase. The entire process is divided into four main steps. Initially, we conduct Ligand Vibration calculations for the Protein-Ligand complex. Then, for each newly calculated conformation of the complex, we compute the Euclidean distances between all protein atom pairs based on their coordinates. These distances are utilized to extract the corresponding potential values from our trained Optimized DrugScore2018 model, thereby forming a Feature Matrix. The final score is determined by summing all potential values within this matrix. Ultimately, only the best score and the associated feature matrix with this top score will be output.
In this process, the calculation of Euclidean distances represents the most computationally intensive step, and thus it is the primary target of our optimization efforts in the following High-Performance Computing (HPC) section.
The Output Feature Matrix. The Feature Matrix, serving as the output of the Optimized DrugScore2018 and the input for the subsequent XGBoost phase, is a crucial component of our methodology. To provide a clearer insight, we present an example of a Feature Matrix as shown in Figure 3.5. This matrix contains 31,875 elements. Due to the large number of elements, the figure only displays 100 protein-ligand atom pairs. For



```
754,P.3_0.3_5.1,0.5
755,P.3_0.3_5.2,1.0
756,P.3_0.3_5.3,0.5
```

```
9990,S.o_0.co2_5.2,1,
9991,S.o_0.co2_5.3,1,
9992,S.o_0.co2_5.4,1,
9993,S.o_0.co2_5.5,1,
9994,S.o_0.co2_5.6,1,
```

Figure 3.3: Illustration of Weighting Schemes: (A) Triangular Weighting Scheme and (B) Full Weighting Scheme for Atom Position Uncertainty

these pairs, all distances of the same atom pair are aggregated, with non-zero components highlighted in blue and zero-valued components left blank. Predominantly, many elements are zero, suggesting that their corresponding atomic pairs don't influence the final outcome. These values represent potential values rather than straightforward scores. Therefore, a more negative value denotes a higher score. For clarity in visual representation, the X-axis of the figure is inverted.

Ligand vibration Technique. For every protein-ligand complex under evaluation, we conducted Ligand vibration. This "Ligand vibration" represents a novel concept in our optimized DrugScore2018 compared to the original. It addresses the previously mentioned atom position uncertainty by simply adjusting the Ligand's position during the inference phase. The adjustment spans 27 directions, as illustrated in Figure 3.6. The black dots indicate the ligand's position in its original conformation. The ligand then shifts to the positions marked by the 6 orange dots (faces), 8 red dots (corners), and 12 green dots (edges), resulting in 26 new conformations. Including the original position, there are a total of 27 conformations. For the moving radius, we treated it as a hyperparameter and conducted 'Try-and-Assess' experiments. As a result, we determined 0.2 Å to be optimal. Detailed comparative analysis can be found in Section 4.1. As mentioned in the previous paragraph, each of these new conformations will be individually evaluated (illustrated in Figure 3.4), only the conformation with the highest score can be selected for output.

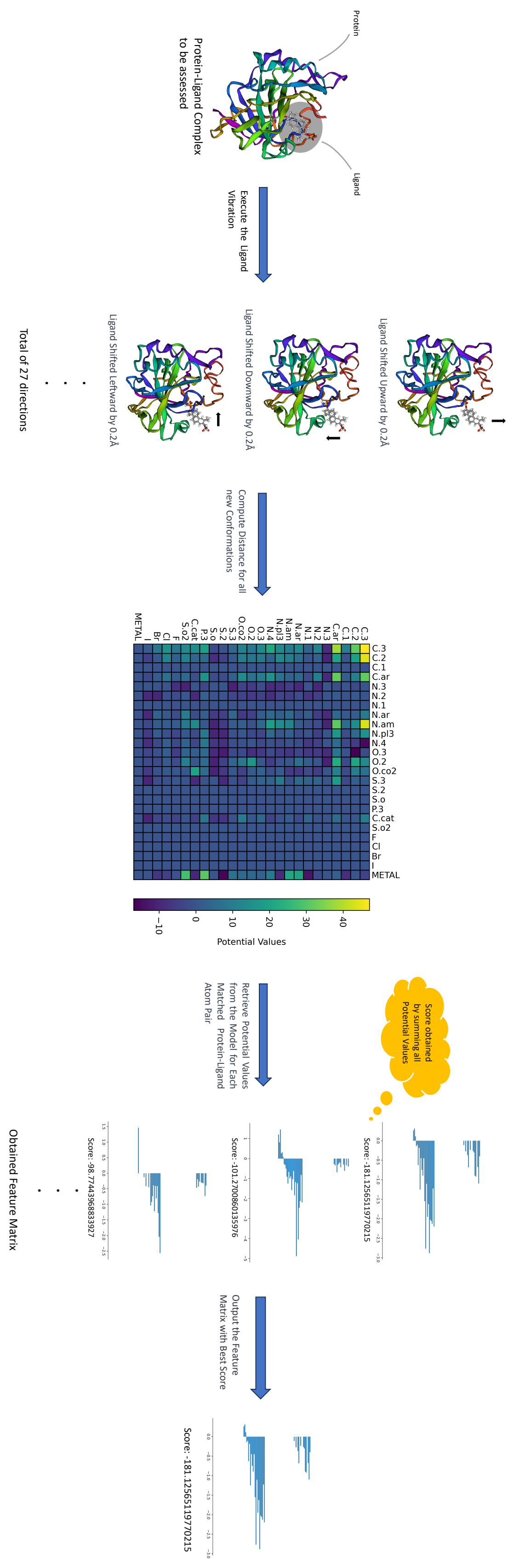

Protein
Ligand
Protein-Ligand Complex to be assessed
Execute the Ligand Vibration
Ligand Shifted Upward by 0.2Å
Ligand Shifted Downward by 0.2Å
Ligand Shifted Leftward by 0.2Å
Total of 27 directions
Compute Distance for all new Conformations
C.3 C.2 C.1 C.ar N.3 N.2 N.1 N.ar N.am N.pl3 N.4 O.3 O.2 O.co2 S.3 S.2 S.o P.3 C.cat S.o2 F Cl Br I METAL
Potential Values
Retrieve Potential Values from the Model for Each Matched Protein-Ligand Atom Pair
Score obtained by summing all Potential Values
Score: -181.1256511970215
Score: -101.27008601335976
Score: -98.77443968833927
Obtained Feature Matrix
Output the Feature Matrix with Best Score
Score: -181.1256511970215

Figure 3.4: Inference Process of Optimized DrugScore2018

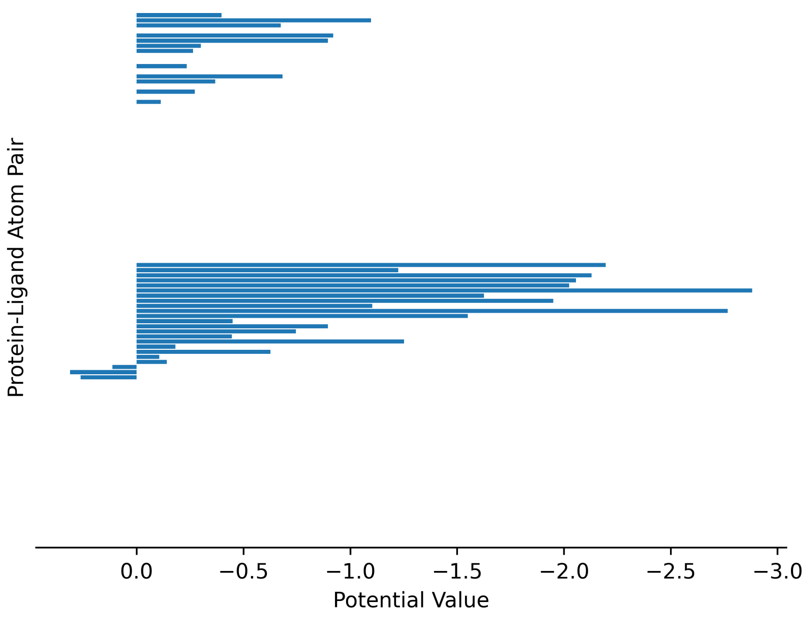


Figure 3.5: Example of XGBoost Feature Matrix

3.3. Training the Machine Learning SF: XGBoost

As outlined in Section 3.1, the inference process of XGBoost leverages the Feature Matrix from Optimized DrugScore2018 as input to predict binding affinity values. In this section, we will focus on the training process of XGBoost, including its data preparation and hyperparameter optimization process.

3.3.1. Data Preparation

For the features utilized in XGBoost, we selected the potential values of protein-ligand atom pairs at various distances. As previously discussed, these features are conveniently generated using the pre-trained Optimized DrugScore2018. The entire workflow for this generation process is depicted in Figure 3.7. This workflow consists of three main steps. Initially, we select and preprocess the Protein-Ligand Complexes data. Next, we input all complex data into Optimized DrugScore2018 to generate the Feature Matrix (a zoom-in example is shown in Figure 3.5). Finally, we compress all these feature matrices into a single Feature Matrix, which forms the final training dataset.
Given the characteristics of XGBoost as a supervised learning method, its approach to data selection and utilization for training significantly differs from the unsupervised

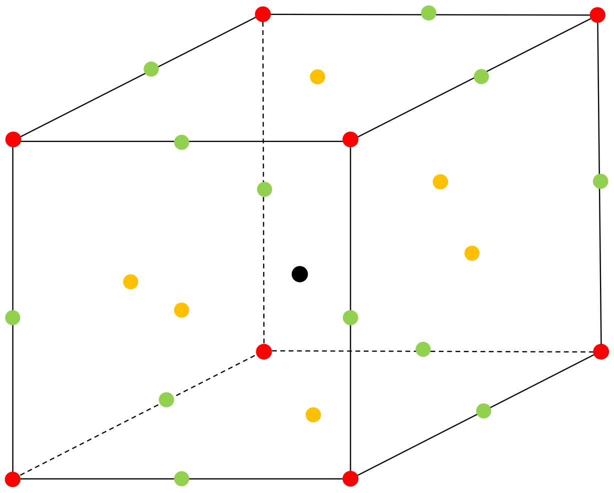


Figure 3.6: The 27 Vibration Directions for the Ligand

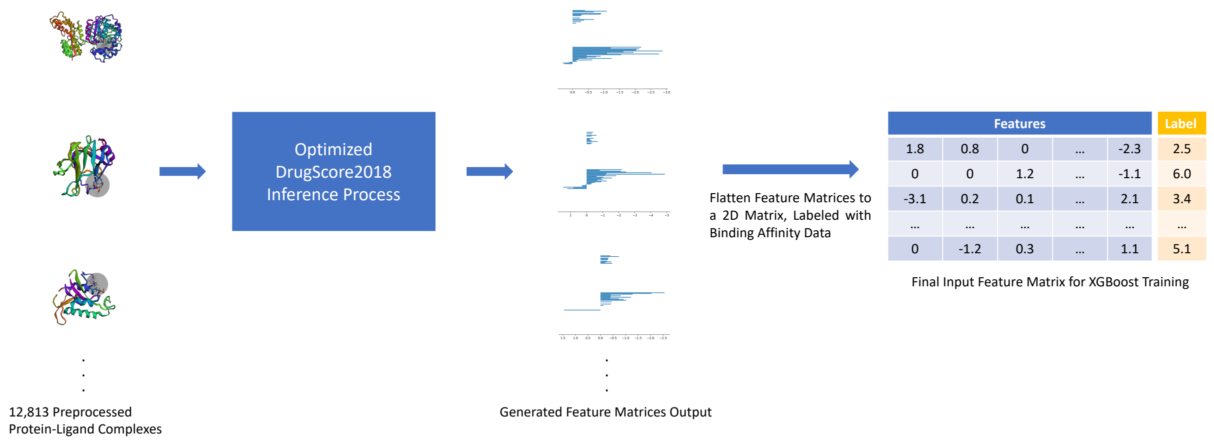
learning-like method used in Optimized DrugScore2018. XGBoost's supervised nature eliminates the need to account for biases from low-affinity ligands, allowing for the use of a more extensive dataset. Therefore, for the XGBoost dataset, we applied just three criteria in contrast to the multiple filtering strategies of Optimized DrugScore2018. Specifically, we:

- Excluded complexes with resolution > 2.5 Å.
- Excluded ligands with < 10 or > 100 heavy atoms (i.e., atoms other than Hydrogen).
- Excluded complexes in the PDBBind coreset (note: the coreset is used as the CASF-2016 test set).

After applying these filtering rules, we ultimately secured 12,813 protein-ligand complexes for the XGBoost training set.

Next, we processed these complexes with our Optimized DrugScore2018's inference mechanism, activating the ligand vibration. In this case, only the potential value matrix with the best score is chosen as the feature matrix for each complex. Incidentally, we also experimented without applying ligand vibration as a control group. Detailed results can be found in Section 4.2.

In the end, the feature matrices from each protein-ligand complex are flattened and merged into a unified 2D matrix, which then serves as the input for XGBoost. Given our dataset of 12,813 protein-ligand complexes, and considering that our Optimized DrugScore2018 model identifies 25 types of protein atoms and 25 types of ligand atoms with 51 distance bins, the dimension of each individual feature matrix is $25 \times 25 \times 51$. As a result, the final input feature matrix has 12,813 rows and 31,875 columns. Each row represents a


12,813 Preprocessed Protein-Ligand Complexes
Optimized DrugScore2018 Inference Process
Generated Feature Matrices Output
Flatten Feature Matrices to a 2D Matrix, Labeled with Binding Affinity Data
Final Input Feature Matrix for XGBoost Training

Features					Label
1.8	0.8	0	...	-2.3	2.5
0	0	1.2	...	-1.1	6.0
-3.1	0.2	0.1	...	2.1	3.4
...	...	...	...	...	...
0	-1.2	0.3	...	1.1	5.1

Figure 3.7: XGBoost Data Preparation Process

protein-ligand complex, while each column indicates the potential value for a specific protein-ligand atom pair at a particular hit distance bin. It's crucial to highlight that many elements within the feature matrix are zeros. These zeros don't denote missing data. Instead, they indicate that certain atom pairs at specific distances have no impact on the final potential, neither positive nor negative. This non-impact is also considered a feature. Moreover, the label for each training complex is derived from the PDBBind index file, using the -logKd/Ki field.

3.3.2. XGBoost Hyperparameter Optimization

Regarding the theoretical underpinnings of XGBoost, including topics like weak learners and tree construction, please consult Section 2.4.2. Here, our emphasis is on the hyperparameter optimization of XGBoost tailored to our specific problem. The primary hyperparameters we optimized for XGBoost include [19]:

- **learning_rate**: This parameter signifies the step size shrinkage. It's used in updates to prevent overfitting. After each boosting iteration, we can directly ascertain the weights of new features. The learning rate reduces these feature weights to ensure a more conservative boosting progression.
- **max_depth**: This refers to the maximum depth of the trees. Making the max depth larger can lead to a more complex model, increasing the risk of overfitting.
- **min_child_weight**: This parameter indicates the minimum sum of instance weight a child node must have. If a tree division results in a leaf node whose total instance weight is below this threshold, then further partitioning is halted. A higher value for this parameter results in a more conservative algorithm.

- **gamma**: This denotes the minimum loss reduction necessary to continue partitioning a tree's leaf node. The higher this parameter is set, the more conservative the algorithm becomes, resulting in fewer tree leaves and a less complex model.
- **subsample**: This parameter denotes the ratio of the samples used for every boosting iteration. It serves to prevent overfitting.
- **colsample_bytree**: This parameter denotes the ratio of the features used for every boosting iteration.
- **reg_alpha**: This represents the L1 regularization on weights.
- **reg_lambda**: This represents the L2 regularization on weights.
- **n_estimators**: The number of the weak learners (trees).

In the loss function, we adopted the squared loss (reg:squarederror), illustrated in Equation 3.1. Here, i represents the training sample number, y_i is the true label, and $\hat{y}_i$ denotes the predicted value. The objective aims to minimize $\mathcal{L}(\theta)$.

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.1}$$

For the aforementioned hyperparameters, we employed Grid Search to optimize them. Define a range for each parameter, test all combinations, and choose the one yielding the lowest loss. For parameter initial ranges, we reference common values or default settings. For instance, if the default value for 'learning_rate' is 0.3, we might initially test it at [0.1, 0.3, 0.5]. Based on the outcomes, the choice can be further refined; if 0.1 yields the best result, we'd then test within the range [0.05, 0.1, 0.15]. This procedure is somewhat tedious, necessitating multiple manual adjustments. The final decision was based on applying the parameter set in practice and evaluating the model's screening power using CASF-2016. Only parameter combinations that yield satisfactory screening power can be ultimately selected.

3.4. Linear Combination of Optimized DrugScore2018 & XGBoost

After sequentially acquiring scores from the aforementioned knowledge-based and machine learning scoring functions, we employed the linear combination approach to derive the final score for DrugXGBScore. Given the differing scales of the knowledge-based and machine learning scoring functions, we applied Max-Min Normalization to the scores from our Optimized DrugScore2018 before combining them, ensuring alignment with the XGBoost scale, as illustrated in Equation 3.2.

$$y_{\text{scaled}} = \frac{y - y_{\min}}{y_{\max} - y_{\min}} \times (12.0 - 2.0) + 2.0 \tag{3.2}$$

The variable y represents the score obtained from the Optimized DrugScore2018. Since these scores are predominantly negative, we convert their sign first to positive to align with the scores derived from XGBoost. The values $y_{\min}$ and $y_{\max}$ denote pre-defined minimum and maximum scores. It's crucial that these preset values accurately represent the underlying characteristics without being excessively extreme. Therefore, we selected them based on the scoring power test results of DrugScore2018. The normalized score, y_{scaled}, is subsequently adjusted to align with the scale utilized by XGBoost, spanning from 2.0 to 12.0.

Subsequently, using the adjustable hyperparameter α and its complement, we achieved a linear combination of the two scores, as illustrated in Equation 3.3.

$$\begin{aligned} \text{DrugXGBScore} = \alpha \times Normalize(\text{OptimizedDrugScore2018}) \\ + (1-\alpha) \times \text{XGBoostScore} \end{aligned} \tag{3.3}$$

We employed Bayesian Optimization to optimize α. This method is a sequential, model-based technique specifically designed for noisy black-box functions [59]. In our approach, we treated the entire scoring function as a black-box, then utilized the CASF-2016 screening power test result, referring only to the average enrichment factor among the top 1%, as the objective function to optimize the parameter α. The specific steps are outlined in Algorithm 3.1. In this procedure, α is the hyperparameter we aim to optimize, y

Algorithm 3.1 Bayesian Optimization Procedure [59]

```
1: for n = 1, 2, 3, ... do
2:     Select $\alpha_{n+1}$ by optimizing the acquisition function $f$:
           $\alpha_{n+1} = \arg\max_{\alpha} \mathbf{f}(\alpha; \mathcal{D}_n)$
3:     Query the objective function to obtain $y_{n+1}$
4:     Augment data with:
           $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\alpha_{n+1}, y_{n+1})\}$
5:     Update the statistical model
6: end for
```

represents the average enrichment factor among the top 1%, while $\mathcal{D}$ denotes the parameter/result set. The acquisition function, $\mathbf{f}$, takes the current α and $\mathcal{D}$ as inputs to guide the optimization process. As demonstrated in Algorithm 3.1, unlike Grid Search, which exhaustively tests all possible combinations, Bayesian Optimization refines its next optimization value using data from previous iterations. This adaptive approach is why we chose Bayesian Optimization.

Figure 3.8 depicts the optimization process. Within it, the Screening Power test and Bayesian Optimization work in tandem, with the efficacy of the Bayesian Optimization hinging on the Power test results. Since α value changes are independent of the optimized drugscore2018 and XGBoost scores (as shown in Equation 3.3) and the CASF-2016 Screening Power test is time-intensive and costly, we input the optimized drugscore and XGBoost results into Bayesian Optimization instead of directly using the final DrugXGBScore. This approach bypasses the need for a full Screening Power test during each α update iteration, significantly reducing optimization time.

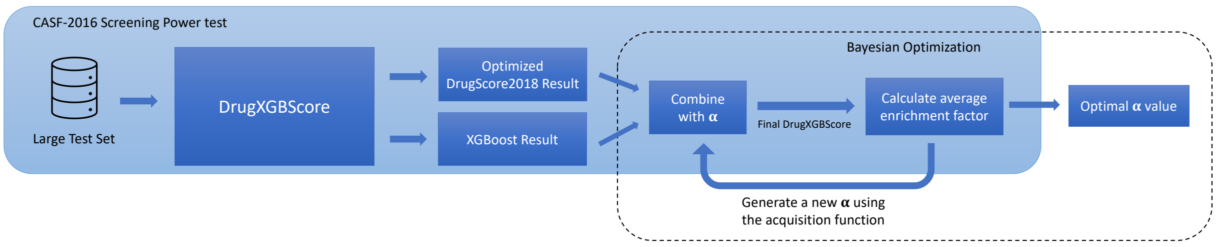

CASF-2016 Screening Power test
Large Test Set
DrugXGBScore
Optimized DrugScore2018 Result
XGBoost Result
Bayesian Optimization
Combine with α
Final DrugXGBScore
Calculate average enrichment factor
Optimal α value
Generate a new α using the acquisition function

Figure 3.8: Combination Parameter Optimization Process

3.5. Deployment on the HPC Pipeline

To boost large-scale drug discovery, we integrated DrugXGBScore into the High-Performance Computing (HPC) pipeline, thereby enhancing high-performance virtual screening. This HPC pipeline is a specialized computational framework designed to handle complex and computing-intensive tasks. It employs advanced computing technologies and hardware, such as parallel computing and high-performance GPUs, to significantly accelerate our overall computing processes. During deployment, we strategically utilized our hardware for optimal parallel computing across both CPU and GPU platforms. We activated all CPU cores to operate independently for maximum efficiency. Concurrently, on the GPU, we tailored our algorithms for optimal execution using CUDA, aligning them with hardware capabilities. These approaches ensure superior computational performance and throughput.

In practice, the input of our HPC pipeline consists of one or more proteins and their corresponding decoys that need to be screened. The output is the predicted score for each decoy, with higher scores indicating a greater binding affinity to the respective protein.

3.5.1. CPU Multi-Threading

Figure 3.9 depicts our CPU multi-threading strategy, wherein the primary thread assigns distinct threads to simultaneously process each protein during virtual screening, free from data dependencies. In cases where the task involves only one protein, it can still be distributed across multiple threads to enable the concurrent screening of various ligands. As outlined in Section 3.1, our choice of XGBoost for the DrugXGBScore was due to its inherent high-performance computing capabilities. Consequently, upon the launch of each individual thread, XGBoost dynamically generates subsidiary threads to meet its computational demands. For our Multithreading Computing Framework, we utilized OpenMP

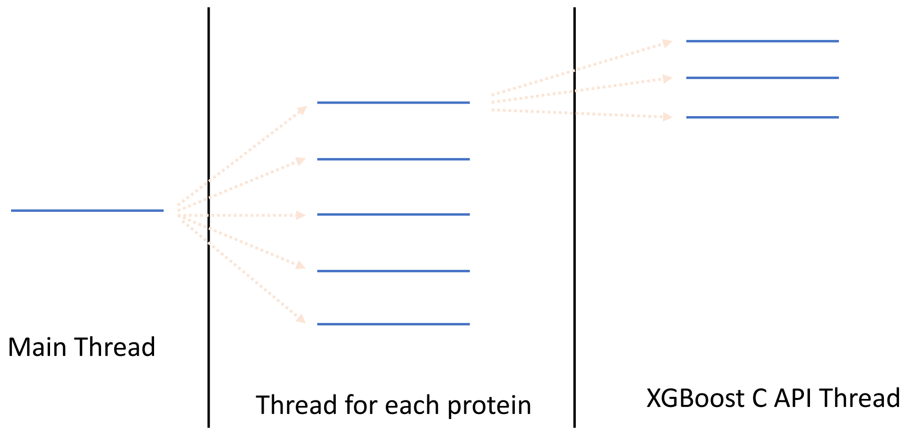


Figure 3.9: CPU Multiple Threads

(Open Multi-Processing) [1], a well-established and user-friendly library with stable performance (refer to Section 1.7.3). Leveraging OpenMP, we launched 18 threads from the main thread on our device, which has 20 CPU cores. This approach was strategically chosen to prevent oversubscription and the resultant thread competition, as XGBoost's autonomous thread management could lead to more than 20 threads vying for CPU time, causing inefficiencies due to excessive context switching.

3.5.2. GPU Acceleration

We enhanced our DrugXGBScore by implementing GPU acceleration with an Nvidia A-100 SXM4 40GB [50]. The NVIDIA A100, based on the advanced Ampere microarchitecture, is purpose-built for artificial intelligence, data analytics, and high-performance computing, making it ideally suited for our computational requirements. Figure 3.10

visualizes the entire GPU accelerating process, using a single thread and protein as a representative example. This process is consistent across all other threads. In this figure, ‘host’ denotes the CPU and its memory, tasked with executing the main program, handling memory transfers, and launching the CUDA kernels. ‘Device’ refers to the GPU and its memory, which are dedicated to running the kernels. Our optimized DrugScore2018

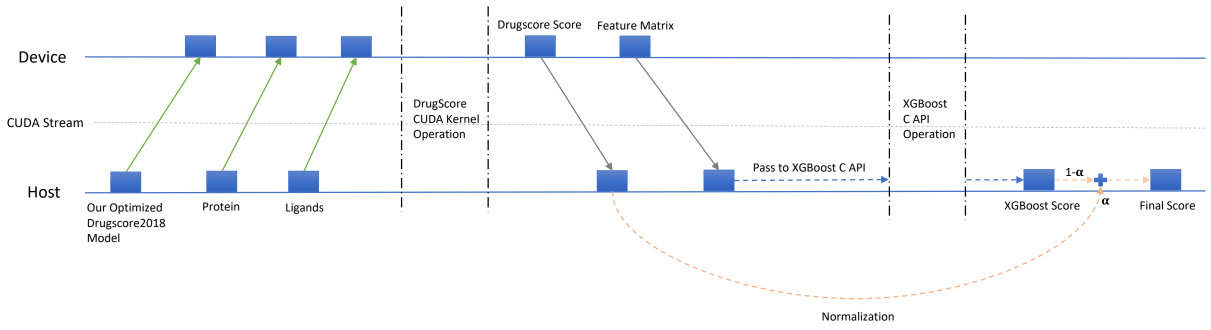


Figure 3.10: GPU Accelerating Process for a single CUDA stream

model and proteins are initially transferred to GPU memory, followed by the ligands awaiting screening. The kernel is then invoked to compute the DrugScore and generate its associated feature matrix. After computation, the data are transferred back to the host memory, where the feature matrix is passed to the XGBoost C API. Finally, the resulting scores are linearly combined with the normalized DrugScore results (refer to Section 3.4). To maximize efficiency, we endeavored to perform all computing-intensive tasks within the CUDA kernel while minimizing memory transfer between host and device. Furthermore, as depicted by the dotted line in the center of the figure, we allocated a CUDA stream to each thread, facilitating concurrent execution of CUDA kernels and data transfers.
Figure 3.11 illustrates the CUDA kernel computational workflow. Given the large volume of ligands awaiting screening, individual processing would lead to suboptimal performance, while simultaneous screening of all ligands is computationally unfeasible. Hence, we adopt batch processing, concatenating the atomic matrices of ligands within a batch into an extended matrix for kernel computation. Subsequently, the kernel calculates distances between ligand atoms and protein atoms, extracts potential values from our trained DrugScore model, and populates the corresponding feature matrix. Ultimately, the score for each ligand in the batch is derived using a block reduction technique that aggregates the potential values within each feature matrix to yield the respective score.
The procedure depicted in Figure 3.11 involves two kernel invocations: the first to obtain feature matrices, and the second to execute block reduction. The implementation of the first kernel invocation is depicted in Figure 3.12, where each curved arrow represents a GPU thread. These threads individually calculate the distances between protein/ligand

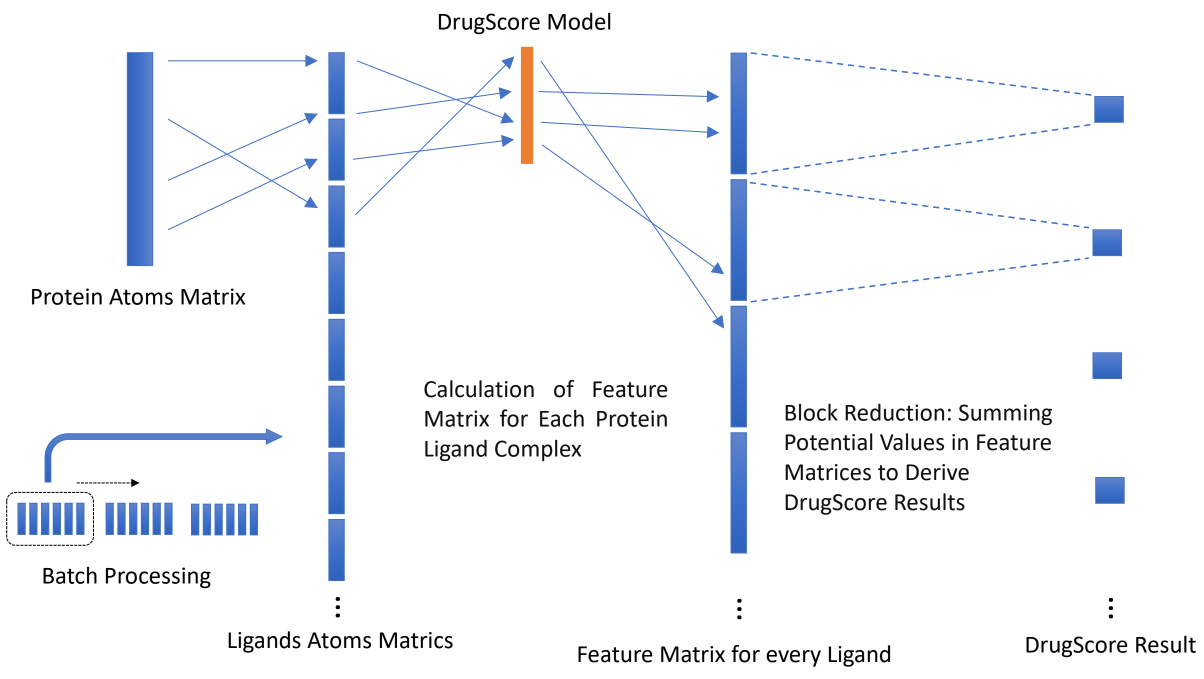

DrugScore Model
Protein Atoms Matrix
Batch Processing
Ligands Atoms Matrics
Calculation of Feature Matrix for Each Protein Ligand Complex
Feature Matrix for every Ligand
Block Reduction: Summing Potential Values in Feature Matrices to Derive DrugScore Results
DrugScore Result

Figure 3.11: CUDA Kernel Calculation Process

atomic pairs, retrieve potential values based on hit distances, and populate the Feature Matrix accordingly. The figure illustrates that we utilize two-dimensional grids and blocks, with the x-axis corresponding to the atoms of each ligand and the y-axis to the atoms of the protein. Block size is determined by the GPU's capacity, and grid dimensions are calculated by dividing the total number of ligand/protein atoms by the block size. Notably, the calculation for the total number of blocks may yield a non-integer value. Since C++ inherently truncates fractions during division, this can result in an insufficient block count. To address this, we adopt the standard solution of incrementing the block count by one, hence the formula: $\lceil \frac{\text{Ligand Rows}}{\text{Block Size}} \rceil$ and $\lceil \frac{\text{Protein Rows}}{\text{Block Size}} \rceil$, where $\lceil \cdot \rceil$ denotes the ceiling function that rounds up to the nearest integer.

Figure 3.13 presents the implementation of the second CUDA kernel invocation, which employs block reduction to condense extended matrices. Our task involves compressing a concatenated matrix of feature matrices into a compact matrix of scores. We determine the grid's X-dimension by dividing the length of each feature matrix by the block size, while the Y-dimension is set to the number of ligands. During execution, each thread computes its value autonomously. After the reduction, the first thread in each block records the block's aggregate to the appropriate location in global memory.

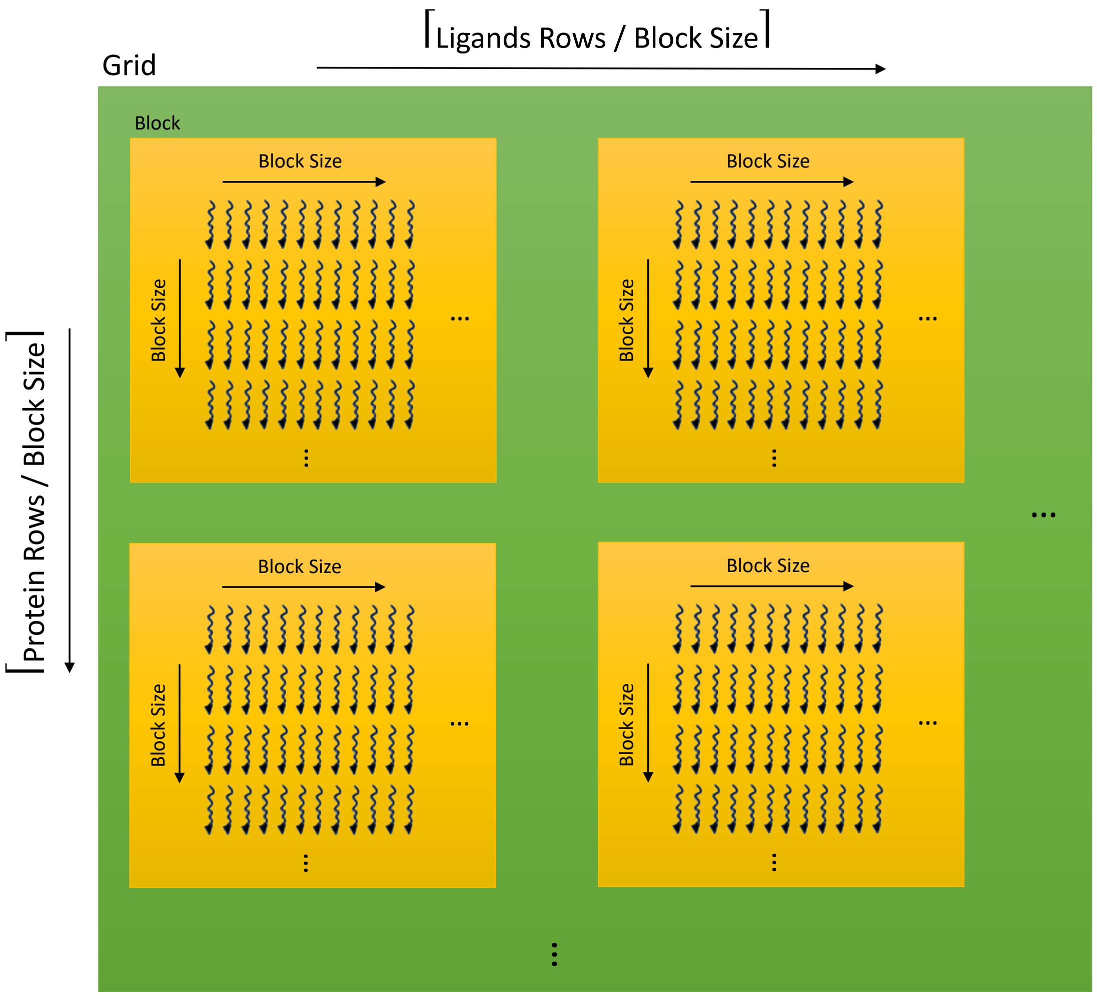


Figure 3.12: Implementation of the Kernel Invocation for Feature Matrix

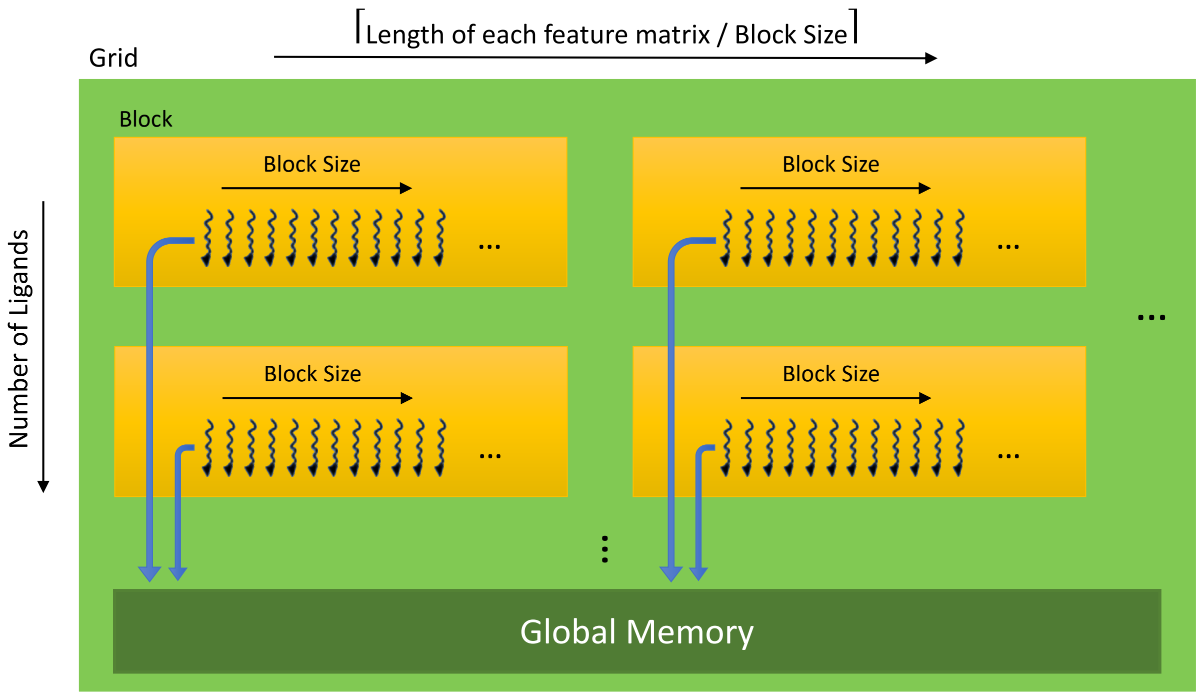


Figure 3.13: CUDA Kernel Block Reduction

3.5.3. Special optimization for Ligand Vibration

In Section 3.2, we explored the optimization of DrugScore2018 through the innovative Ligand Vibration technique. Distinct from other methods, such as parameter optimization, this technique incurs computational overheads, potentially impacting computing performance in the prediction stage. To mitigate this, we developed a specialized CUDA implementation aimed at minimizing the performance impact of this technology.
Figure 3.14 illustrates the CUDA kernel operation applied to the original Ligand Batch Matrix, expanding it by computing the new coordinates of each ligand, and storing these in an expanded matrix. As depicted in Figure 3.6, for our optimized DrugScore2018, each ligand is vibrated in 26 directions, including the original coordinates, yielding 27 distinct conformations. Consequently, the expanded matrix is 27 times the size of the original.

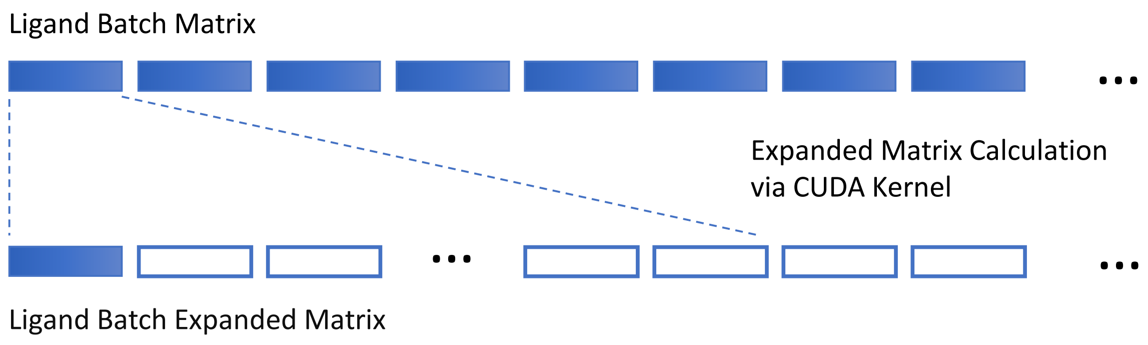

Ligand Batch Matrix
Expanded Matrix Calculation via CUDA Kernel
Ligand Batch Expanded Matrix

Figure 3.14: Schematic Diagram of Ligand Vibration Matrix Expansion

To efficiently manage the expanded matrix, we applied specific configurations. Considering drugs are typically small molecules, we standardized all ligands to a uniform length, such as 100. As discussed in Section 3.2, the DrugScore2018 training dataset excluded ligands with more than 100 heavy atoms, making a standardized length of approximately 100 reasonable. However, this standardization is flexible and can be adjusted depending on the ligands being screened. Ensuring uniform ligand matrix lengths means that the extended matrix for each batch remains consistent in size, which allows us to preallocate GPU memory space before batch processing. This approach facilitates the reuse of the extended matrix across various batches, thereby avoiding the computational overhead from repeated memory allocations and deallocations.
The CUDA kernel implementation for expanding the ligand vibration matrix, illustrated in Figure 3.15, employs a one-dimensional configuration. The grid dimension is set to match the number of ligands per batch, with each block dedicated to processing a single ligand. Meanwhile, the block dimension corresponds to the number of atoms within a lig-

and, ensuring that each thread is tasked with computing and placing the new coordinates of a single atom, based on the vibration direction, into the appropriate location within the expanded matrix. This implementation is straightforward and resource-efficient. With the 'Number of Ligands in Batch' and 'Number of Atoms in Ligand' being modest, it will require only limited GPU resources. By operating with CUDA streams, this kernel can execute concurrently with other kernels, facilitating kernel-level overlapping in the GPU pipeline.

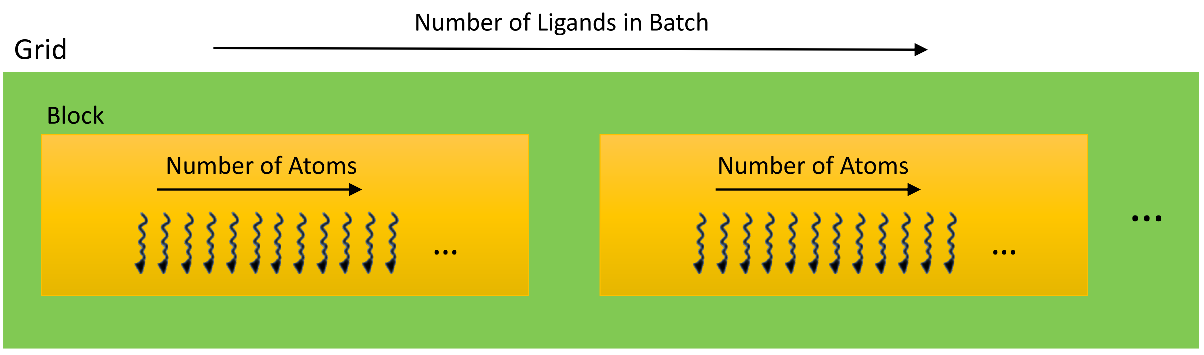


Figure 3.15: CUDA Kernel Implementation for Ligand Vibration Matrix Expansion

4 | Experimental results

In this chapter, we detail the experimental results for each component of our proposed methodology and provide relevant discussions. We present not only the positive final outcomes but also compare them with control group results to highlight the efficacy of our specific choices. We will separately discuss the DrugXGBScore and its layout results within the HPC pipeline. Within DrugXGBScore, we employ a bottom-up approach, initially highlighting the enhancements of Optimized DrugScore2018 and XGBoost before revealing the overall DrugXGBScore outcome.
We used CASF-2016 [63] to evaluate the predictive accuracy of our DrugXGBScore. As detailed in Section 1.6, CASF-2016 comprises four tests: Scoring Power, Ranking Power, Docking Power, and Screening Power. The Screening Power test is notably the most resource-intensive, followed by the Docking Power test, with the Scoring and Ranking Power tests being comparatively lightweight. Our main reference is the outcome of the Screening Power test, but the Scoring and Ranking Power test results provide preliminary insights. While a minor underperformance in the Scoring/Ranking Power test doesn't necessarily indicate poor Docking/Screening Power outcomes, if a component completely fails the Scoring/Ranking Power test, it's prudent to skip the Docking/Screening Power test to save computational resources and time.

4.1. Optimized DrugScore2018 Results

In this section, we delve into the experimental test results and analysis for the Optimized DrugScore2018, corresponding to the proposal detailed in Section 3.2. Our focus encompasses the selection of specific technical details and the results from the CASF-2016 Power Test.
Data Selection Analysis. First, we assessed our Optimized DrugScore2018 data selection process using the CASF-2016 Screening Power tests. As described in Section 3.2, we applied five exclusion strategies and further refined our selection from the filtered sets with different sizes based on their predictive power. Among our filtering strategies, three were previously validated by Dittrich et al. [21]: excluding complexes with a resolution

greater than 2.5, excluding complexes in the PDBBind coreset (used by CASF-2016 as a test set), and excluding ligands with fewer than 10 or more than 100 heavy atoms. This eliminated the need for re-testing these strategies. After applying them, 10,088 complexes remained. Additional filtering of ligands with binding affinity values where -logKd/Ki is less than 6.0 brought the number down to 8,637. Lastly, removing ligands with a Tanimoto Similarity value greater than 0.2 and a frequency above 200 left a final count of 6,048 complexes. Subsequently, we sampled subsets from the filtered 6,048 complexes and assessed the Screening Power of the DrugScore2018 model trained on them under consistent conditions. Notably, during this sampling, we consistently gave priority to complexes from the PDBbind "refined set".

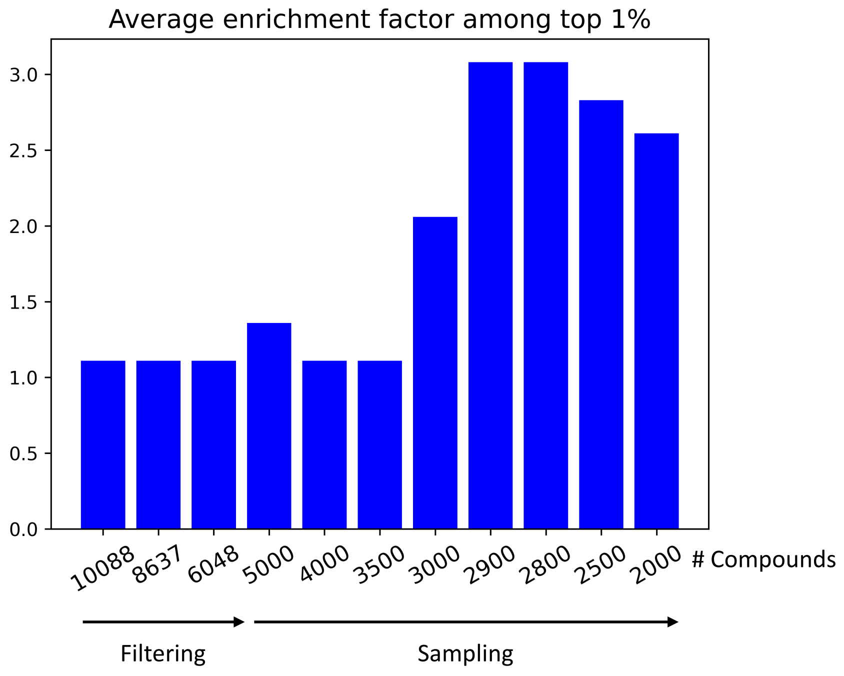


Figure 4.1: Screening Power of Models Trained Using Corresponding Selection Sets

The results are presented in Figure 4.1. The x-axis represents the size of the selected training set, while the y-axis depicts the average enrichment factor among the top 1% for the corresponding model. From the results, it's evident that the model performs optimally when the training set size is between 2800 and 2900. We ultimately selected 2900 to maximize the number of complexes used for training while ensuring the best outcome. Of the 2,900 complexes, 1,146 are sourced from the PDBbind "refined set", with the remaining 1,754 derived from the PDBbind "general set". Additionally, as observed in Figure 4.1, results for sets with fewer than 2800 or more than 3000 complexes show significant underperformance. We theorize that sets with fewer than 2800 complexes may not provide sufficient data for model training, leading to under-saturation. Conversely, for sets larger than 3000, incorporating a greater number of complexes from the lesser-quality PDBbind "general set", in comparison to the PDBbind "refined set", considerably weakens the screening power.

Smoothing Function Analysis. As detailed in Section 3.2, we implemented two smoothing functions in the DrugScore2018 model to address atom position uncertainty: the Triangular Weighting Scheme and the Full Weighting Scheme, as illustrated in Figure 3.3. The test results are depicted in Figure 4.2. When compared under identical

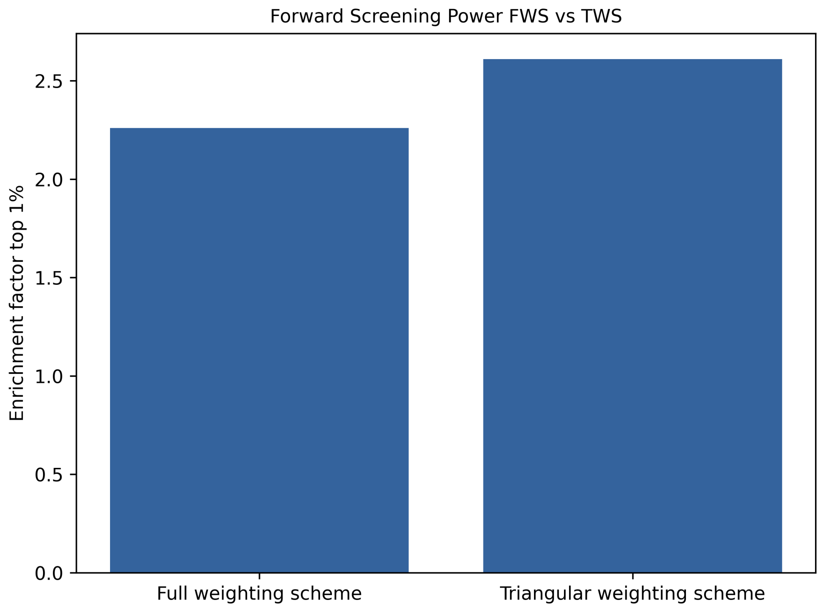


Figure 4.2: TWS vs FWS

conditions, the milder Triangular Weighting Scheme (TWS) exhibits higher Screening Power than the more aggressive Full Weighting Scheme (FWS). The average enrichment factor among the top 1% improves by 0.35. Given the sizable test set for the Screening Power, even a 0.01 improvement is noteworthy without substantially impacting computational efficiency. Importantly, while TWS and FWS differ in the training phase, their computational performances are identical during inference. Therefore, with the same computational performance but improved prediction accuracy, TWS proves to be more advantageous.

Final Optimized DrugScore2018 Model Analysis. After the detailed data selection, preprocessing, and application of the Triangular Weighting Scheme, the occurrences of our finalized model are presented on the left side of Figure 4.3. Note that such figures aggregate all distance bins for a given atom pair. According to Velec et al. [67]'s findings, potentials derived from over 500 pair interactions ensure sufficiently smooth and reliable

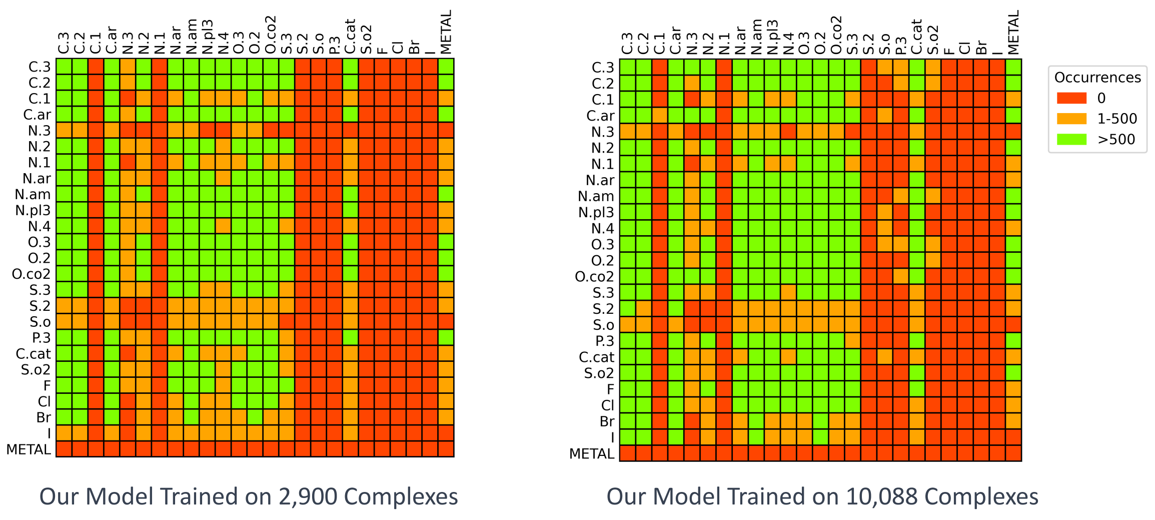


Figure 4.3: Our Model Occurrences

results. Our model, trained on 2,900 complexes (Figure 4.3 Left), encompasses 525 pair interactions. Of these, 206 interactions have occurrences exceeding 500, while 133 fall between 1 and 500. For reference, Figure 4.3 (Right) depicts a model trained using the entire dataset of 10,088 complexes before our filtering, boasting 249 pair interactions with occurrences exceeding 500. While the occurrences in this model are seemingly superior to the final model, the increased noise in this training set prevented it from outperforming the one trained on 2,900 complexes in the Screen Power test, as shown in Figure 4.1. Similarly, we present the results for DrugScore Original [28] and DrugScore2018 [21] in Figure 4.4 for comparison. Utilizing the same color scheme as in Figure 4.3, red in

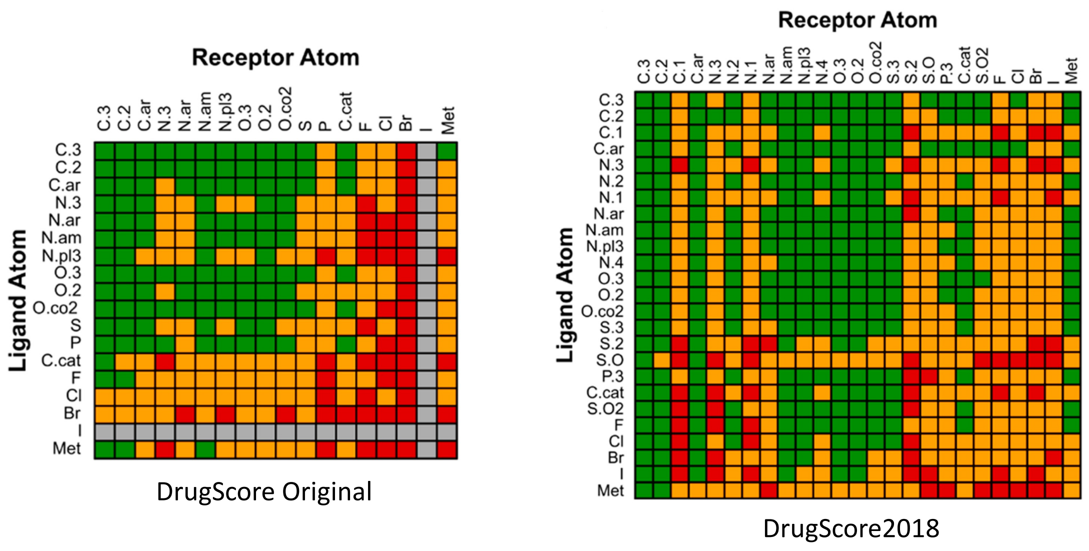


Figure 4.4: Occurrences in DrugScore as reported by [28] and [21]

these figures indicates a Number of Occurrences equal to 0, orange represents occurrences between 1 and 500, and green signifies occurrences greater than 500. Due to the use of different datasets, the Number of Occurrences in our model differs from that in DrugScore. Our final model's number of green blocks lies between that of DrugScore Original, which has 117 pair interactions exceeding 500, and DrugScore2018, with 289 pair interactions exceeding 500. These observed differences might primarily stem from the data set quality. Enhancing this quality or obtaining higher-quality data could substantially boost our model's prediction accuracy. Additionally, in Chapter 5, we also discussed several future directions for further improving data set quality.

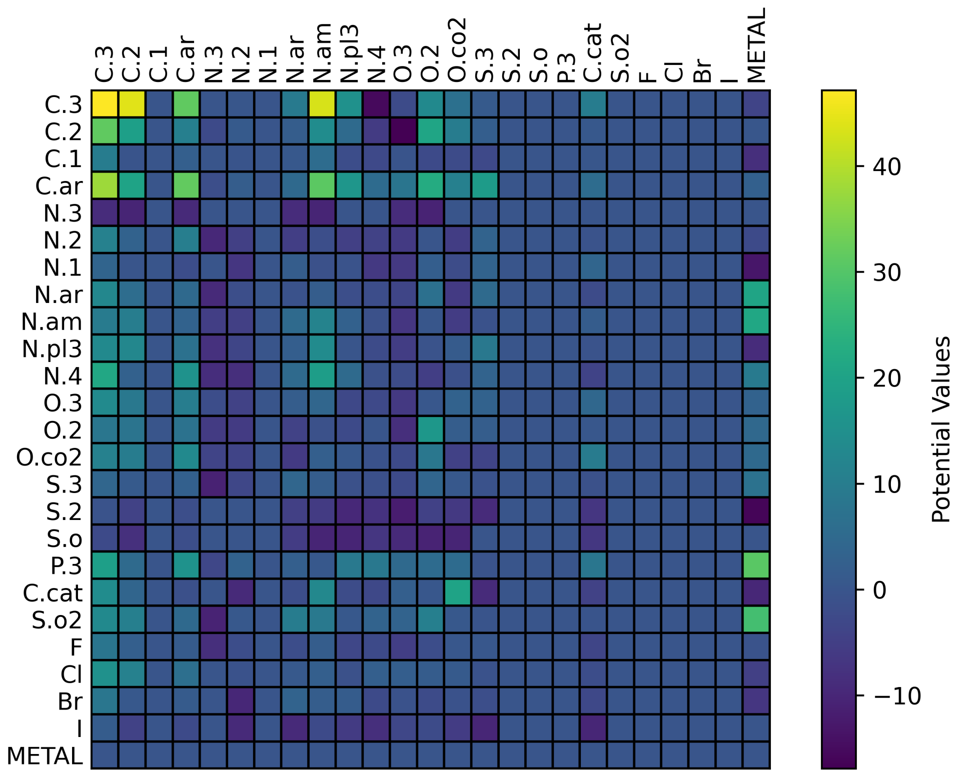


Figure 4.5: Pairwise Potential of Our Final Model

The pairwise potential compiled from the occurrences is depicted in Figure 4.5. This pairwise potential can be understood as our final Optimized DrugScore2018 model. As detailed in Section 3.2, during the inference stage, we simply extract the potential value from this model based on the hit atom pair with their Euclidean distance. Given that this model represents potential values rather than mere scores, a more negative value signifies a stronger positive influence on the final result. As illustrated in Figure 4.5, darker colors indicate superior potential values, while lighter shades denote inferior potential.

Ligand Vibration Technique Analysis. As described in Section 3.2, we employed the ligand vibration technique during the inference stage to manage the uncertainty in protein-ligand conformations. This approach also partially mitigates issues related to our dataset quality. While moving the ligand in 27 directions around its original position, we considered the movement radius as a hyperparameter. We then evaluated the screening power for each radius and selected the one that exhibited the highest screening power as our final choice. The results are depicted in Figure 4.6. A vibration radius of 0.2 yields the best Screening Power. This aligns, to some degree, with the findings of Kossiakoff et al. [41] and Gohlke et al. [28], suggesting that for a resolution of 2.5 Å, atom position inaccuracies can be up to 0.4 Å.
In Figure 4.6, compared to the control group "No Vibration", there's a noticeable enhancement in Screening Power for vibration radii between 0.15 and 0.25, with a peak at 0.2. However, unlike the cost-free improvement seen with Triangular Weighting Scheme and Full Weighting Scheme, Ligand Vibration introduces a significant computational expense. Still, as discussed in Section 3.5.3, we've optimized the use of hardware for the Ligand Vibration Technique, allowing its integration into our Optimized DrugScore2018 with minimal overhead.

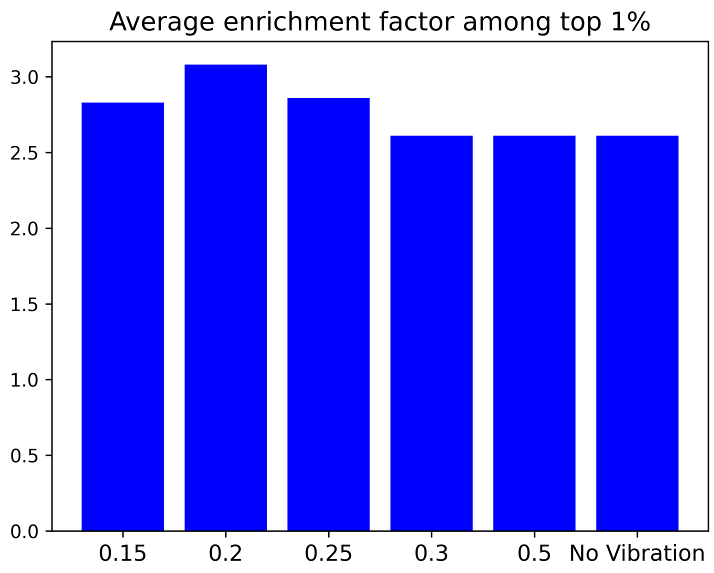


Figure 4.6: Ligand Vibration with Specific Radius

Optimized DrugScore2018 Power Test Result. After employing CASF-2016 Screening Power Test to finalize the aforementioned techniques and parameters, we conducted a comprehensive evaluation of the final Optimized DrugScore2018 using CASF-2016. We primarily used DrugScore2018 as the baseline, aiming to achieve results comparable to it.

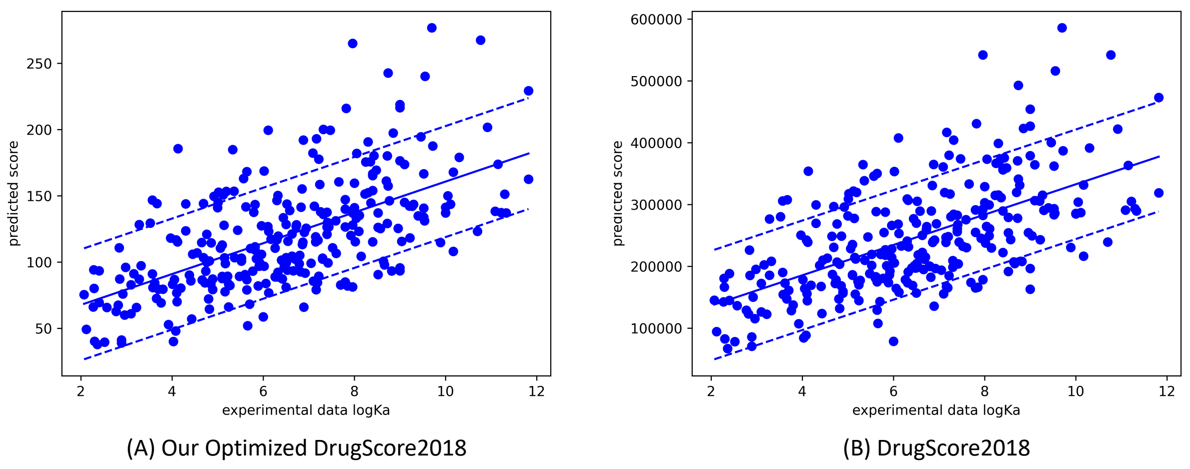


Figure 4.7: Scatter Plot Comparison

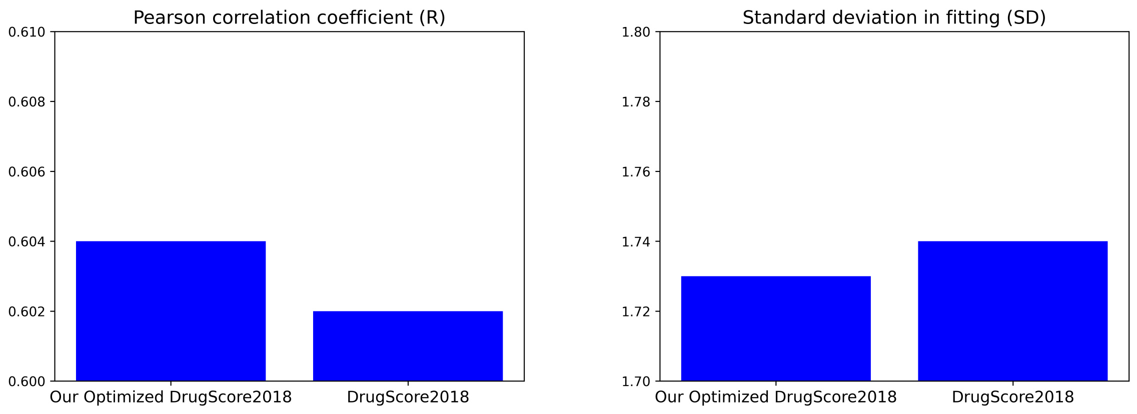


Figure 4.8: Scoring Power Comparison

See Figures 4.7 and 4.8 for the Scoring Power test results. Figure 4.7 presents a scatter plot that maps the scores from our Optimized DrugScore2018 model and DrugScore2018 against the experimental binding affinities data for the 285 Protein-Ligand Complexes tested, all within a two-dimensional space. The central solid line represents the linear regression line, while the dotted lines above and below depict the regression line adjusted by one standard deviation above and below, respectively. While the scores of the two scoring functions operate on different scales, it's the relative values, rather than the absolute ones, that matter. Thus, they remain comparable in this figure. Figure 4.8 displays the Pearson correlation coefficient (R) and the standard deviation in fitting (SD) for the two scoring functions. A higher Pearson correlation coefficient is desirable, while a lower

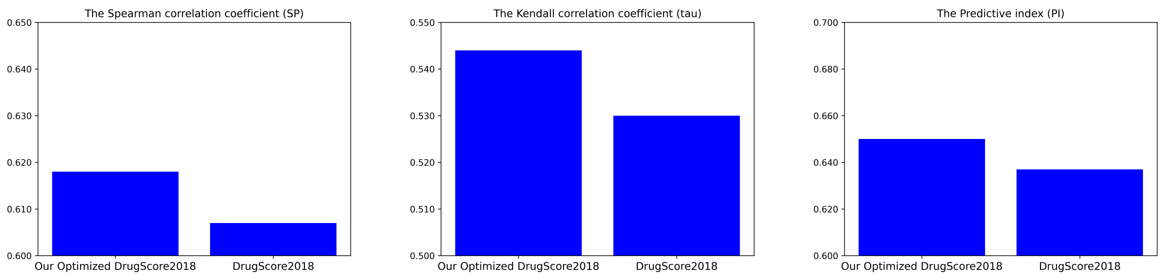


Figure 4.9: Ranking Power Comparison

standard deviation is preferable. Given the similar values achieved by both functions, we adjusted the y-axis range in the figure to emphasize their differences. From these two figures, it's evident that the scoring power of our Optimized DrugScore2018 slightly surpasses that of DrugScore2018. However, the difference is negligible. In essence, their performances are virtually identical.

Figure 4.9 presents the results of the Ranking Power test, comparing three metrics: Spearman correlation coefficient (SP), Kendall correlation coefficient (tau), and Predictive Index (PI). Larger values for these metrics indicate better performance. As with previous figures, we also adjusted the y-axis scale here to emphasize the differences between them. The results indicate that our Optimized DrugScore2018 has a slight edge over DrugScore2018 in Ranking Power, with metric differences approximating 1%.

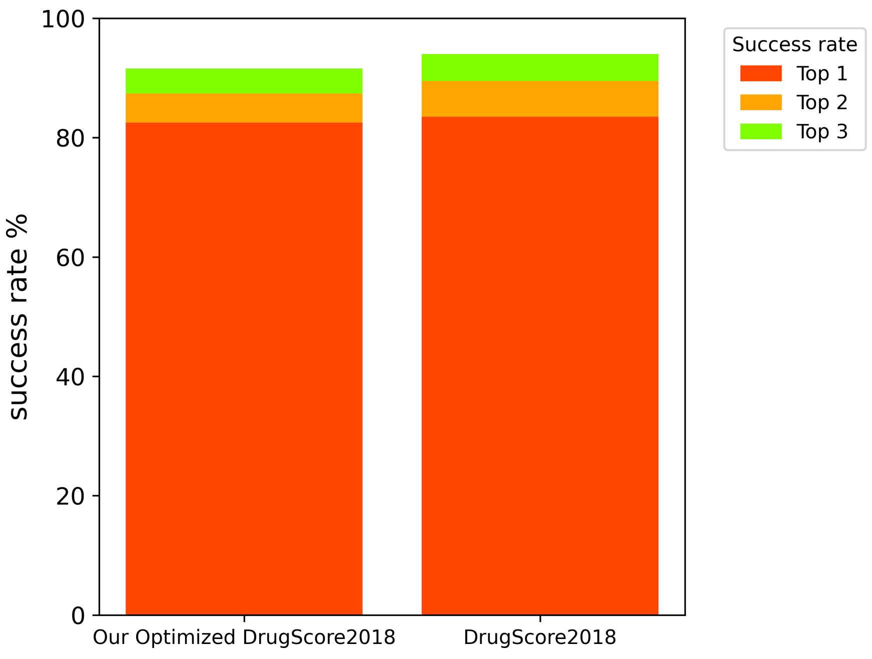


Figure 4.10: Docking Power Comparison

Figure 4.10 presents the Docking Power test results. In this figure, red, orange, and green denote the success rates for the top 1, top 2, and top 3 binding poses, respectively, as ranked by the scoring function. Overall, our Optimized DrugScore2018's docking power closely mirrors that of DrugScore2018, with a difference of less than 1%. Given that the test set for Docking Power is substantially larger than that for Scoring Power, this discrepancy is acceptable.
The Screening Power of our Optimized DrugScore2018 and DrugScore2018 is depicted in Figures 4.11 and 4.12, representing the Forward Screening Power and Reverse Screening Power, respectively. A key metric to consider from these figures is the average enrichment factor for the top 1% in Forward Screening Power, highlighted on the right side of Figure 4.11. Additionally, the success rate for Forward Screening Power is displayed on the left of Figure 4.11. The tri-colored bars indicate the best ligand being identified among the top 1%, 5%, and 10% candidates. Figure 4.12 focuses on Reverse Screening Power, with the colors similarly representing the best target discovery within the top 1%, 5%, and 10% candidates.

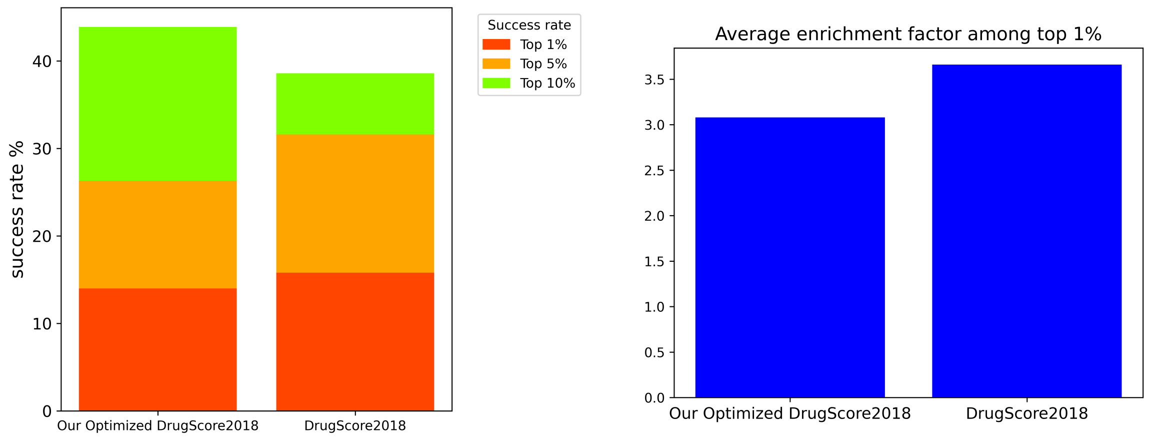


Figure 4.11: Forward Screening Power Comparison

From the results presented in both figures, our Optimized DrugScore2018 demonstrates a success rate closely aligned with DrugScore2018 in both Forward and Reverse Screening Powers. However, a noteworthy divergence is observed in the crucial metric: the average enrichment factor among the top 1%, where we see a difference of 0.58 compared to DrugScore2018. We theorize this discrepancy arises due to the vast size of the Screening Power test set, exposing the quality limitations in our dataset. Fortunately, as evidenced in Sections 4.2 and 4.3, the integration of XGBoost not only bridged this gap but also further enhanced the average enrichment factor of our hybrid scoring function.

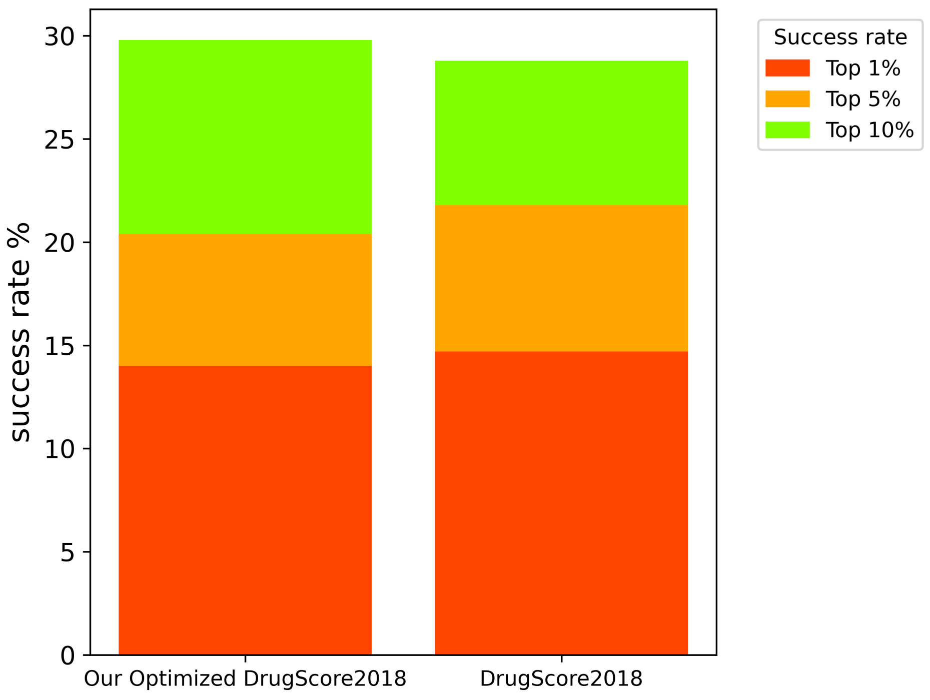


Figure 4.12: Reverse Screening Power Comparison

4.2. XGBoost and Linear Combination Results

In Sections 3.1 and 3.3, we detailed the integration of XGBoost with our Optimized DrugScore2018 and explored its training process. This section shifts our focus to the results and analysis of XGBoost. We will examine the application of the Ligand Vibration Technique within XGBoost, present the findings from the XGBoost Grid Search, and discuss the outcomes of the CASF-2016 Power Test. Its outcomes become significant only when integrated with our Optimized DrugScore2018. Therefore, we utilize the Optimized DrugScore2018 and its direct combination with XGBoost results as a baseline to emphasize the improved outcomes. Notably, we do not consider the combination parameter α (as referenced in Equation 3.3) within this discussion.

Ligand Vibration Technique in XGBoost. Similar to our Optimized DrugScore2018, we also employed the Ligand Vibration Technique in XGBoost. However, while Optimized DrugScore2018 incorporates it during the inference stage, XGBoost uses the feature set from the output of vibrated DrugScore2018 for model training, applying the Ligand Vibration Technique during the training phase. This approach offers a clear advantage: it avoids using such a computing-intensive technique in inference, eliminating unnecessary computational costs.

As illustrated in Figure 4.13 A and B, the XGBoost model trained with the vibrated

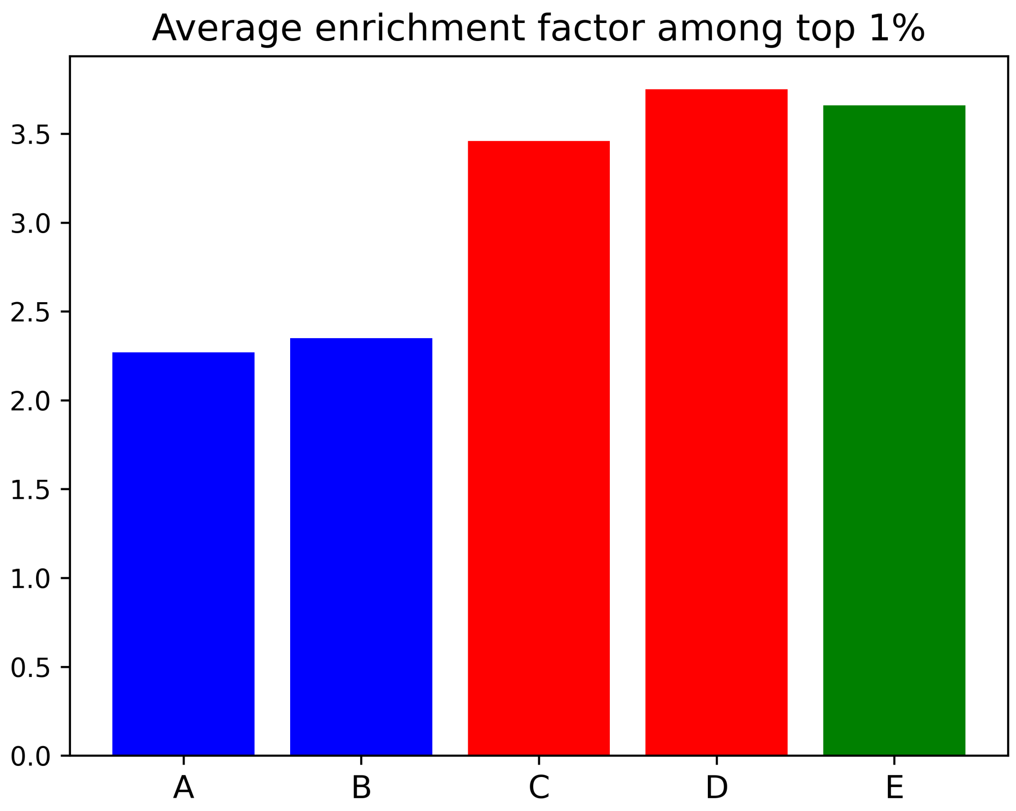


Figure 4.13: **XGBoost Screening Power Comparison: Considering Vibration Effects.** (A) XGBoost without Ligand Vibration. (B) XGBoost with Ligand Vibration. (C) Combined Optimized DrugScore2018 and XGBoost, excluding Ligand Vibration. (D) Combined Optimized DrugScore2018 and XGBoost, including Ligand Vibration. (E) DrugScore2018.

feature set marginally outperforms the non-vibrated counterpart. Though the enhancement is modest, it represents a significant improvement given that there's no additional computational cost during the inference phase. When integrated with our Optimized DrugScore2018, the enhancement from Ligand Vibration becomes evident (refer to Figure 4.13 C and D). Remarkably, the combined result surpasses the screening power of DrugScore2018 (see Figure 4.13 E), mitigating, to some extent, the inherent limitations of our dataset's quality.

XGBoost Grid Search Analysis. As discussed in Section 3.3, we employed grid search to determine the best hyperparameters for XGBoost. Table 4.1 specifically lists the defined parameter ranges, culminating in 19,683 potential configurations. When subjected to three-fold cross-validation, this gave rise to 59,049 model evaluations.

Parameter	Range of Values
colsample_bytree	0.7, 0.8, 0.9
gamma	0.0, 0.05, 0.1
learning_rate	0.03, 0.05, 0.07
max_depth	8, 9, 10
min_child_weight	5, 6, 7
n_estimators	500, 550, 600
subsample	0.7, 0.8, 0.9
reg_alpha	0.0, 0.05, 0.1
reg_lambda	0.0, 0.05, 0.1

Table 4.1: Range of Parameters for XGBoost Grid Search

However, exploring every configuration was impractical, it was also unnecessary due to the independence of some parameters. By testing them individually, we significantly cut down our experimental efforts. After several attempts, we pinpointed the optimal configuration, which we present below. This configuration achieved a Root Mean Square Error (RMSE) of 1.426.

- colsample_bytree=0.8
- gamma=0.05
- learning_rate=0.05
- max_depth=10
- min_child_weight=6
- n_estimators=600
- reg_alpha=0
- reg_lambda=0.05
- subsample=0.8

XGBoost Power Test Analysis. Figure 4.14 is the Scoring Power Scatter Plot of the XGBoost model. Based on the visualization in the figure, XGBoost's scoring power exhibits a marked improvement compared to DrugScore2018 (refer to Figure 4.7). The majority of points lie close to the solid regression line, with only a few outliers beyond the bounds of the regression line adjusted for standard deviation.

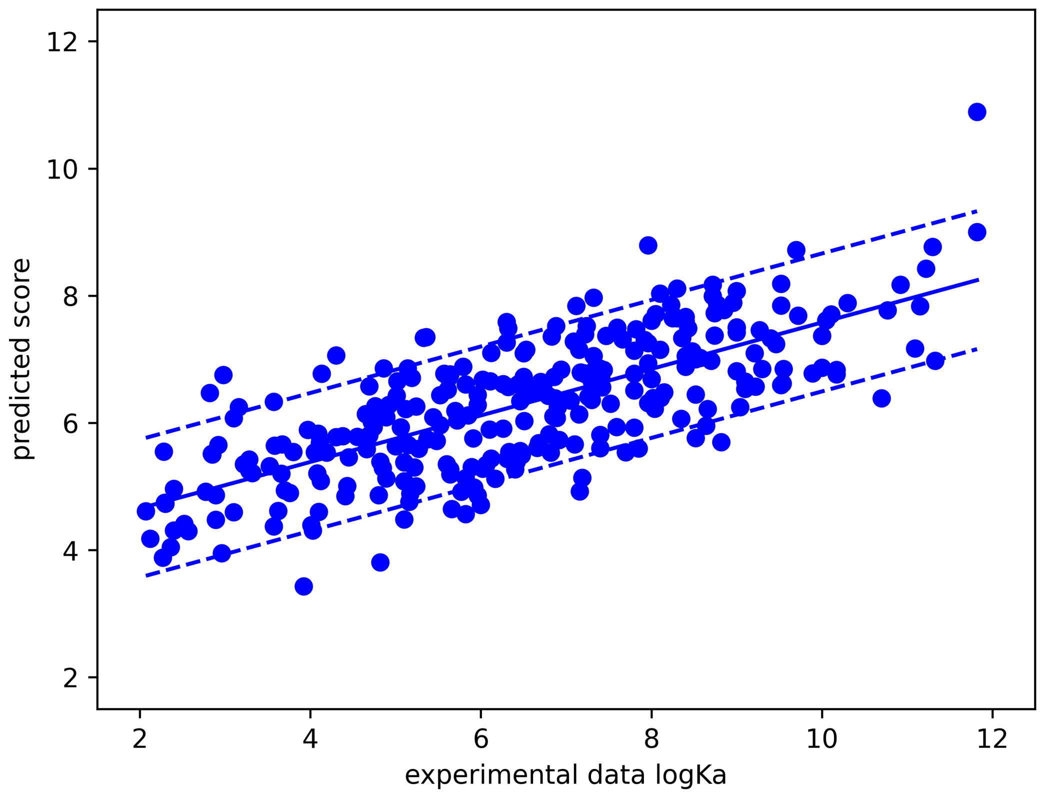


Figure 4.14: Scoring Power Scatter Plot for XGBoost Model

In Figure 4.15, we present the CASF-2016 Power Test results for the XGBoost model. Within each subfigure: (A) represents our Optimized DrugScore2018 results, (B) showcases the XGBoost results, (C) depicts its direct combination with DrugScore2018, and (D) displays the DrugScore2018, which serves as the control group. The figure shows that the integration of DrugScore2018 with XGBoost enhanced performance in Scoring, Ranking, and Screening Power tests. Although Docking Power doesn't see a significant boost, it yields results consistent with our Optimized DrugScore2018. Of these, XGBoost's Scoring Power stands out, with its Pearson correlation coefficient (R) reaching 0.733. This score surpasses most Scoring Functions provided in CASF2016 (refer to the comparison in Section 4.3).

While XGBoost's marked improvement in Scoring Power is commendable, it's not entirely positive. This may suggest that we've trained XGBoost more towards Scoring Power than Screening Power, our primary concern. The loss function of XGBoost, used in both grid search and training, resembles the squared error in Equation 3.1, inherently aligned with Scoring Power.

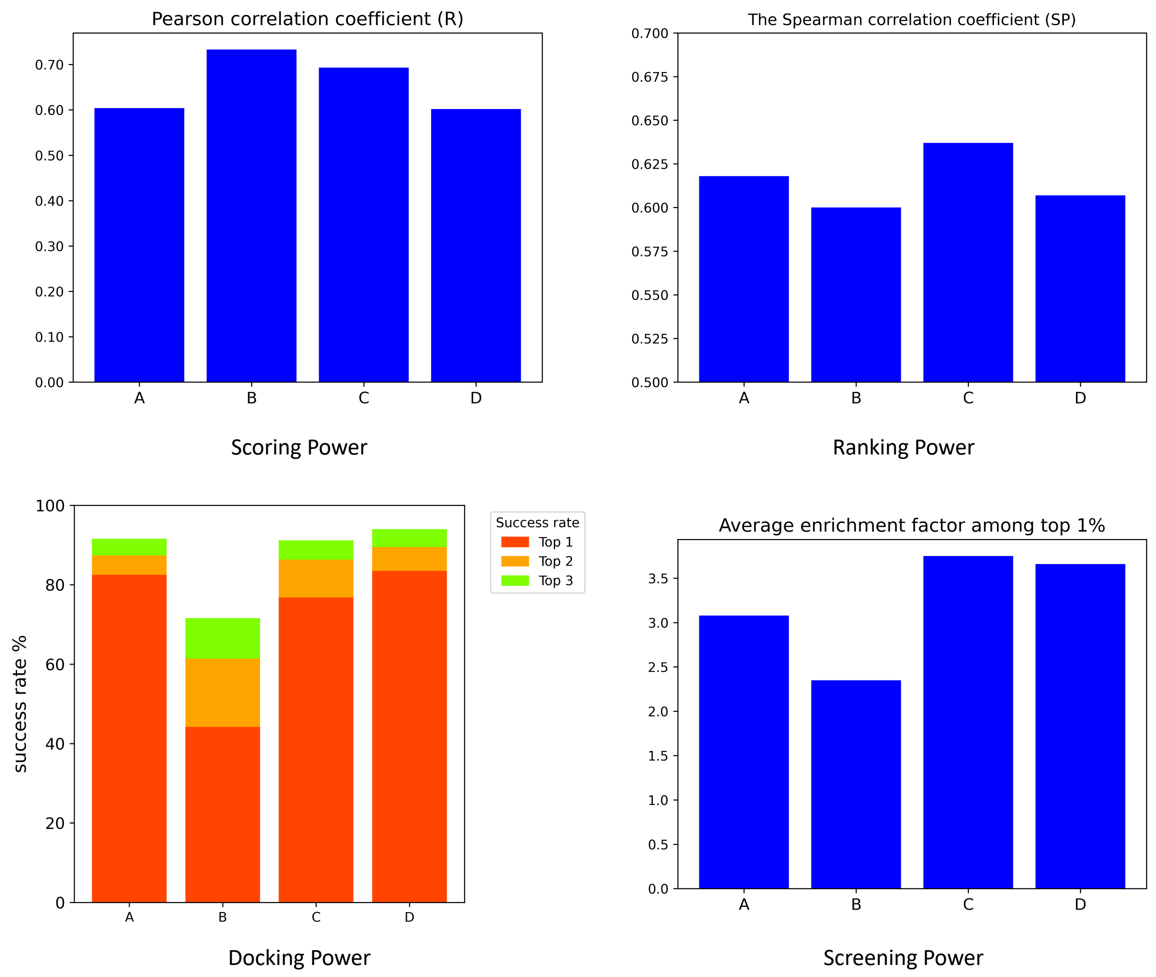


Figure 4.15: **XGBoost Power Test Result.** (A) Optimized DrugScore2018. (B) XGBoost. (C) Combined Optimized DrugScore2018 and XGBoost. (D) DrugScore2018.

4.3. Overall DrugXGBScore Outcomes

In this section, we evaluate our proposed DrugXGBScore. Using the parameter alpha from Equation 3.3, we combined the Optimized DrugScore2018 with the XGBoost model. Subsequently, we performed a Power Test on DrugXGBScore and compared its performance with other Scoring Functions highlighted in CASF-2016.

Bayesian Optimization Analysis. In Section 3.4, we discussed the use of Bayesian Optimization to identify the optimal value for α. Figure 4.16 illustrates these results. On the x-axis, we have the α values, and the y-axis shows the corresponding target, defined as the average enrichment factor among the top 1%.

As the outcomes of 30 iterations presented in Figure 4.16, the optimal α value consistently hovered around 0.37, and we ultimately chose 0.37051443623449176 as the final

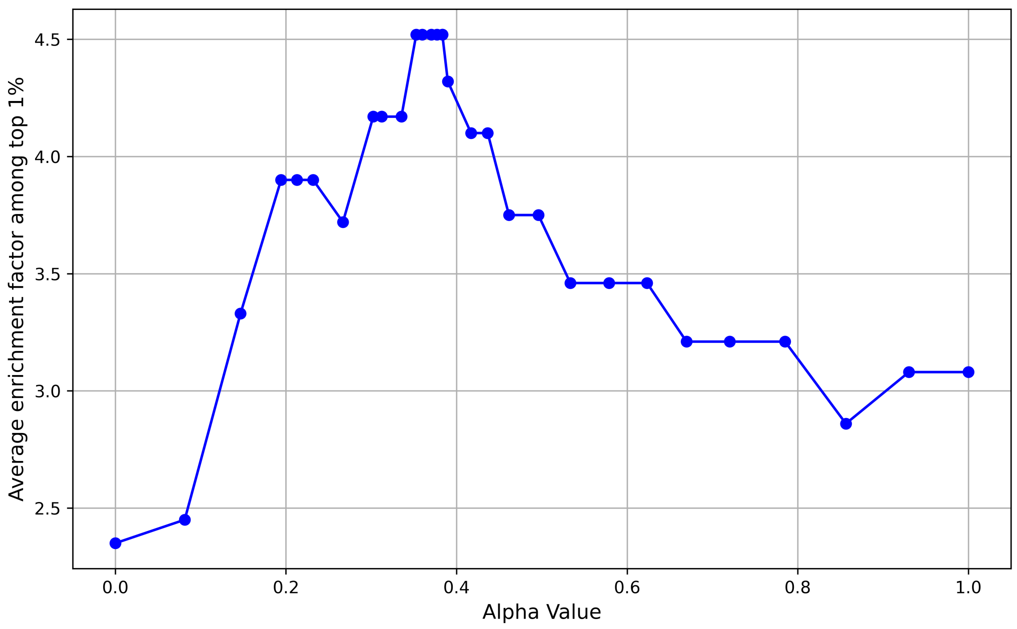


Figure 4.16: Bayesian Optimization Result

result. In this context, an α of 0 signifies exclusive reliance on Optimized DrugScore2018, while a value of 1 indicates sole dependence on XGBoost. The optimal solution's leftward inclination suggests a greater contribution from Optimized DrugScore2018 to the final score. Consequently, our final reference result's average enrichment factor among the top 1% improved from 3.75, as noted with the direct combination (equivalent α set to 0.5) in Section 4.2, to 4.52.

Comparison of DrugXGBScore with Other Scoring Functions. Figure 4.17 presents the results of our DrugXGBScore Power test, alongside comparisons with other example scoring functions provided by CASF-2016, listed in order. In the four subgraphs, our Optimized DrugScore2018 appears in green, XGBoost in orange, the final DrugXGBScore in red, and the other Scoring Functions in blue. The first row displays Scoring Power and Ranking Power, and the right of the second row presents Screening Power. In line with our previous approach, we use the Pearson correlation coefficient (R), Spearman correlation coefficient (SP), and the Average enrichment factor for the top 1% as respective reference target values. Meanwhile, on the left of the second row, Docking Power showcases success rates for the top 1, top 2, and top 3 binding poses. The color gradation reflects these rates, with the darkest shade denoting Top1 and the lightest indicating Top3.

From the data presented in Figure 4.17, our DrugXGBScore (highlighted in red) achieves near-optimal performance in Scoring and Ranking Power and ranks in the upper-middle tier for Docking and Screening Power. We believe this result aligns well with the prediction accuracy requirements for this drug discovery task.

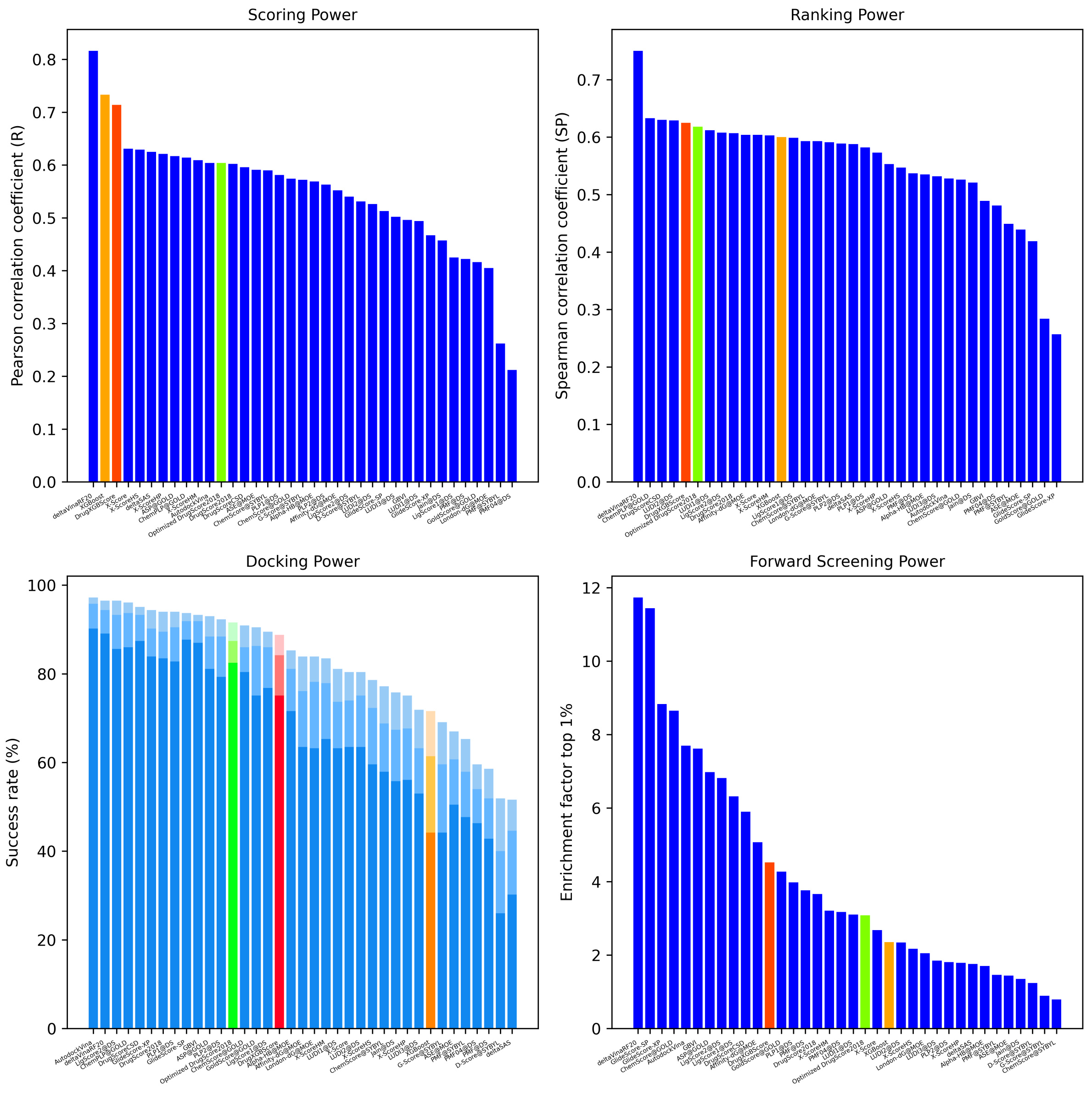


Figure 4.17: **Comparison of DrugXGBScore with Other Scoring Functions.** Key: Green - Optimized DrugScore2018; Orange - XGBoost; Red - DrugXGBScore; Blue - Other scoring functions.

4.4. HPC Implementation Outcomes

In this section, we detail the acceleration impact achieved by integrating DrugXGBScore into our custom-designed HPC pipeline, with a focus on the enhancements in computational performance. We adopted a bottom-up approach to analyze the performance improvements resulting from specific technique trade-offs during the HPC pipeline construction. Subsequently, we assess the overall performance enhancements achieved through GPU acceleration in comparison to CPU-only usage. For the evaluation dataset, we utilized the off-the-shelf CASF-2016 Screening Power benchmark, which includes 57 target proteins, each accompanied by 28,500 decoys (ligands), amounting to a total of 1,624,500 decoys for screening. This extensive test set served as the basis for assessing the computational performance of our DrugXGBScore, with throughput and total running time as the principal metrics for evaluation.

Analysis of Simultaneous Thread Execution. In Section 3.5, we noted that our CPU comprises 20 cores, and to prevent oversubscription due to additional threads generated by XGBoost, we strategically launched only 18 threads concurrently. Consequently, we conducted a test using all proteins and their corresponding decoys from the Screening Power dataset, comparing the overall running time when launching 18 and 20 threads concurrently.

The result presented in Figure 4.18 indicates that launching 18 or 20 threads simul-

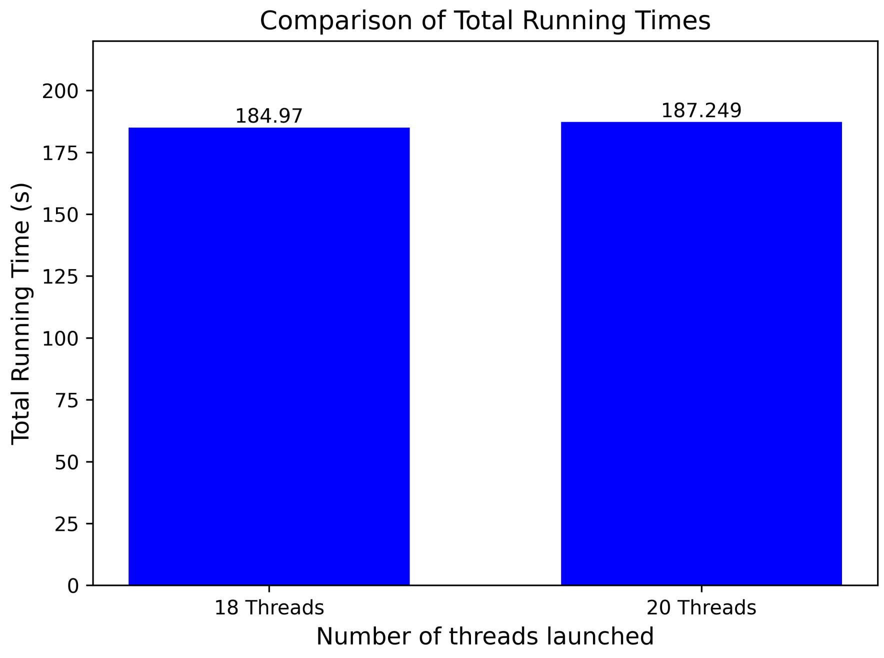


Figure 4.18: Result of CPU Threads Test

taneously exerts a negligible influence on the total screening time. However, given the substantial size of the dataset, the observed enhancement of approximately three seconds is also noteworthy. Importantly, the thread count for this operation can be dynamically tuned to suit the capabilities of a particular device. This experiment underscores the critical need to address the oversubscription issue associated with XGBoost. To optimize performance, it is advisable to set the thread launch count to approximately 10% less than the total number of available CPU cores.

Analysis of the Impact of Ligand Vibration Technology on Computing Performance. As outlined in Section 3.2, the Ligand Vibration Technique was adopted to mitigate uncertainty in protein-ligand conformations, thereby enhancing dataset quality. While this technique inherently increases computing cost, Section 3.5.3 details how we have refined hardware utilization to integrate this technique into our Optimized DrugScore2018, effectively minimizing the additional overhead. To assess the effectiveness of our optimizations and the extent to which the Ligand Vibration Technique affects computing performance, we conducted a controlled experiment. This evaluation was carried out under uniform conditions to directly compare scenarios without the technique to those where it was applied post-hardware optimization.

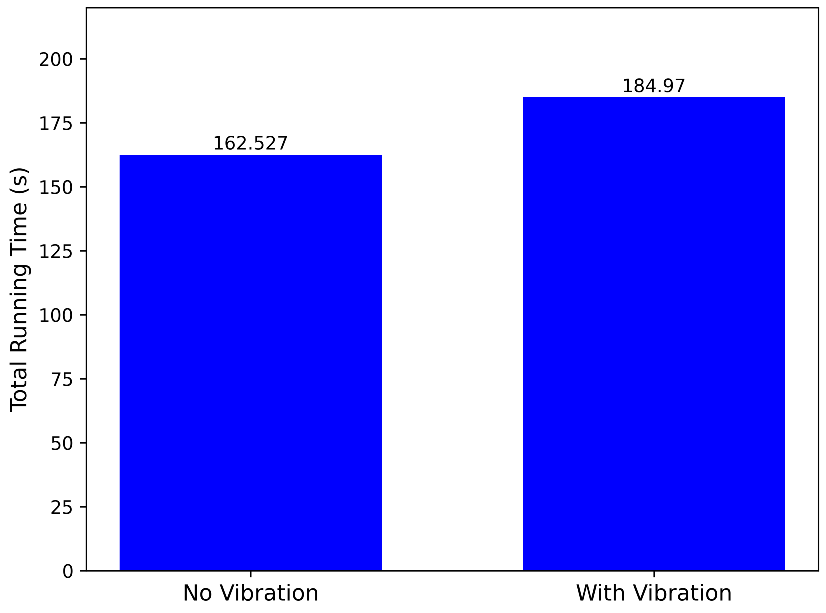


Figure 4.19: Performance Impact of the Ligand Vibration Technique

The results shown in Figure 4.19 demonstrate that the Ligand Vibration Technique will impact overall performance despite optimization efforts. This effect is attributed to calculating 27 vibration directions and the subsequent prediction for all of them. However, our project's objective extends beyond computing performance impacts to striking an optimal balance between prediction accuracy and computational efficiency. In this screening test, which encompassed 57 proteins and 1,624,500 decoys, the application of the Ligand Vibration Technique led to an increase in processing time of only 20 seconds, rather than multiplying the total running time by 27. This modest increase indicates the effectiveness of our optimization efforts. Considering the enhancement in prediction accuracy, as demonstrated in Figure 4.6, this additional time is deemed a justifiable and acceptable computational expense. Moreover, Ligand Vibration functions can be treated as an independent component within our DrugXGBScore HPC pipeline. Its activation can be tailored based on specific application scenarios and requirements.

NVIDIA Nsight Systems Visual Evaluation. After finalizing the technical specifications, we advanced to a comprehensive evaluation of the DrugXGBScore's performance throughout the entire HPC pipeline. To gain insights into the internal execution dynamics of our pipeline, NVIDIA Nsight Systems was utilized for an in-depth visual analysis. This allowed us to determine whether our pipeline can effectively implement parallel computing between threads and streams.

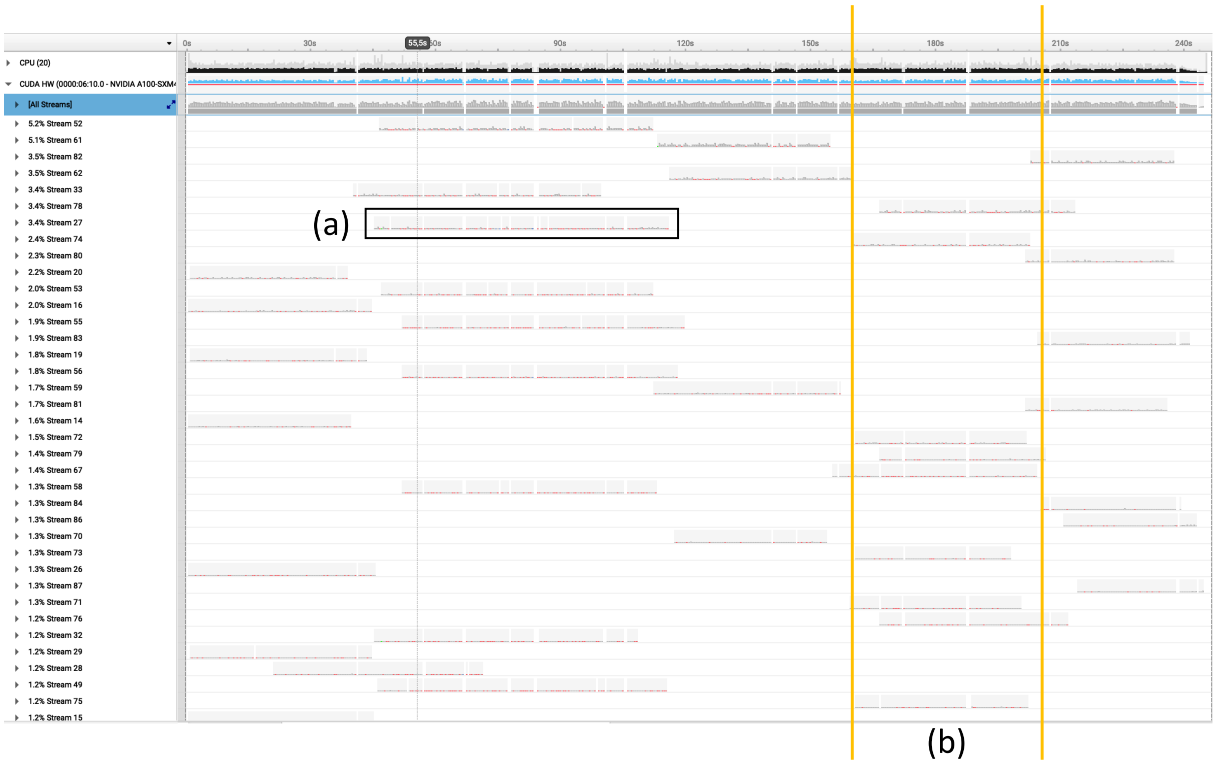
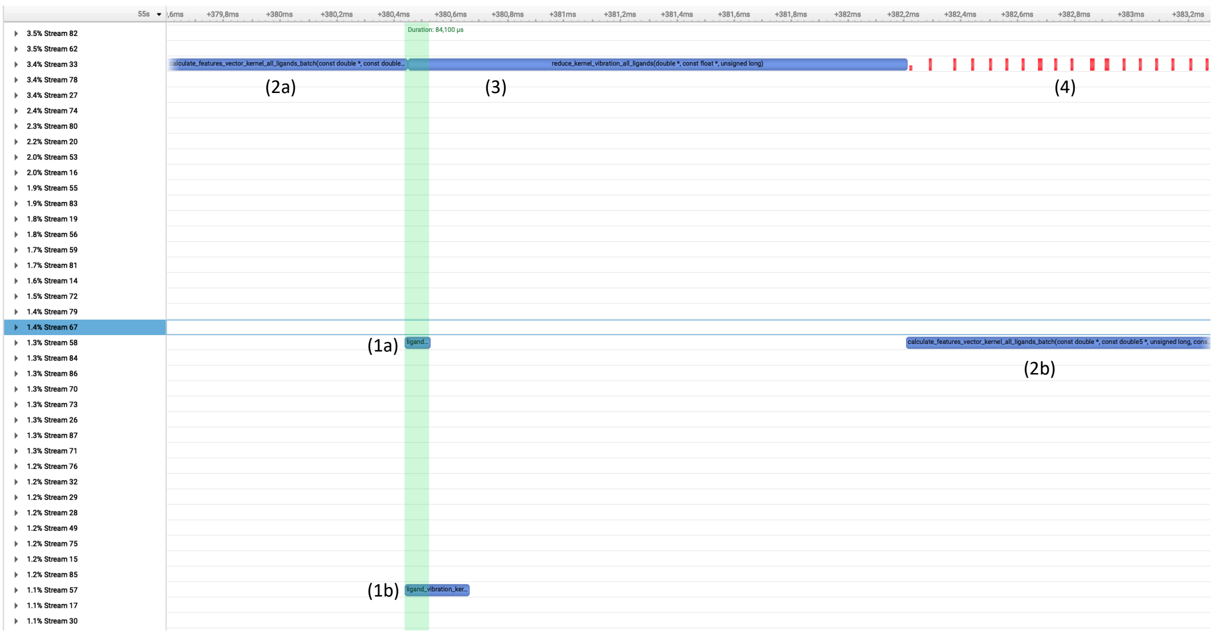


Figure 4.20: Visual Report Generated by NVIDIA Nsight Systems

Figure 4.20 is an overview of the program execution, captured from NVIDIA Nsight Systems' visual evaluation report. As shown in this figure, the program's entire execution took approximately 240 seconds. Running the program in Debug Mode is necessary to generate this report, so the total runtime exceeds that of Release Mode. The left side of the figure depicts all the CUDA Streams. Our execution logic assigns a non-default Stream to each thread, resulting in a total of 57 non-default Streams corresponding to the 57 Proteins. Additionally, each thread is provided with a default Stream for handling operations such as CUDA Malloc, Memory Set, and CUDA Free. The shaded area in the figure represents the execution status of each Stream. For instance, as indicated by label ‘a’, it encompasses both the kernel call and the data transfer operation. The overlapping of kernel and memory operations across different Streams, as exemplified by the yellow-lined area labeled ‘b’, demonstrates the effectiveness of parallel computing in this implementation. In this specific area, at least 9 streams are executing operations concurrently.



55s
Duration: 84,100 µs
calculate_features_vector_kernel_all_ligands_batch(const double *, const double...
reduce_kernel_vibration_all_ligands(double *, const float *, unsigned long)
(2a)
(3)
(4)
(1a)
ligand...
calculate_features_vector_kernel_all_ligands_batch(const double *, const double5 *, unsigned long, cons...
(2b)
(1b)
ligand_vibration_ker...

Figure 4.21: Overlapping Detail in Visual Report

Figure 4.21 provides an in-depth view of the parallel computations. It displays three Kernel instances, each serving a distinct purpose. Kernel 1 is dedicated to calculating the Ligand Vibration matrix. Kernel 2 computes the mutual distances between protein-ligand atoms, subsequently integrating these values into the feature matrix as outlined in Figure 3.11. Kernel 3 is responsible for Block Reduction. This figure notably illustrates a typical instance of Kernel overlapping within the green area, where Kernels 1, 2, and 3 in different Streams partially coincide. Kernel 2, being the most computation-intensive, shows minimal overlap with other Kernels, in contrast to the more substantial overlapping observed between Kernels 2 and 3. Additionally, the figure also highlights the parallelism between memory operations and kernel executions. The red block, marked as ‘4’, rep-

resents the CUDA asynchronous memory copy operation, which is overlapped with our most computation-intensive Kernel 2 for optimized performance.
In summary, the visual report from Nvidia Nsight System reveals that our HPC implementation of DrugXGBScore achieved fundamental stream-level parallel computing, with kernels and memory operations overlapping effectively to fulfill high-performance computing objectives.
Comparative Analysis of GPU Acceleration versus CPU-Only Usage. Subsequently, we conducted a performance evaluation of DrugXGBScore in a CPU-only setup. This was compared with the total runtime following GPU acceleration under identical conditions, allowing us to assess the extent of performance improvement achieved by the HPC pipelines.
To this end, we conducted two distinct sets of experiments to comprehensively evaluate the performance of DrugXGBScore under various scenarios.

- In the first set, we utilized all 57 proteins for screening, along with their corresponding 1,624,500 ligands.
- For the second set, our focus was on screening potential decoys for a single protein, ACETYLCHOLINESTERASE, which has 8313 atoms, is identified by PDB Code 1E66, and includes 28,500 decoys.

Additionally, as previously discussed, Ligand Vibration can be considered an independent component. To demonstrate our HPC pipeline's performance across different application scenarios, we conducted tests on the two aforementioned sets under two conditions: with and without activating the Ligand Vibration Technique.

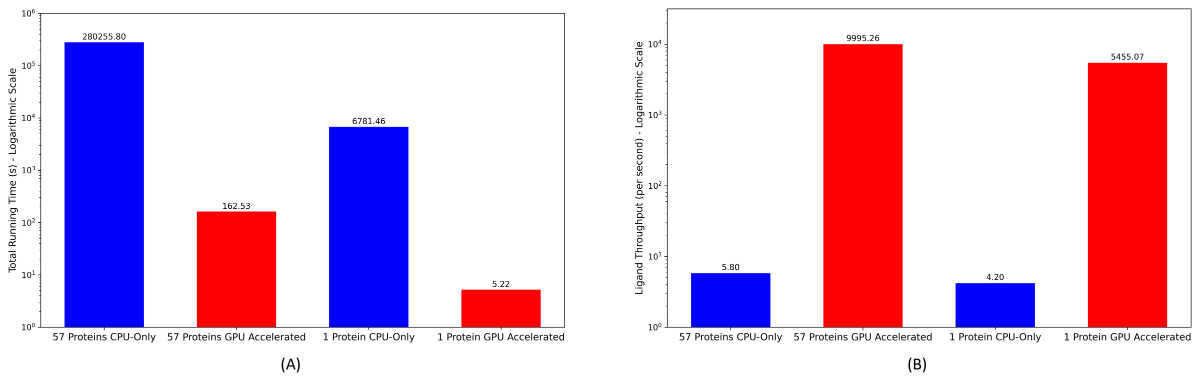


Figure 4.22: Comparative Analysis of our HPC Pipeline Performance without Ligand Vibration

The results of the DrugXGBScore experiments, conducted without the Ligand Vibration Technique, are displayed in Figure 4.22. Panel A illustrates the total running time, while

Panel B shows the ligand throughput per second. The blue bars represent the CPU-only runs for both 57 proteins and ACETYLCHOLINESTERASE, whereas the red bars denote the GPU-accelerated runs for these entities. Due to the substantial runtime difference between CPU-only and GPU-accelerated runs, a Logarithmic Scale was utilized for the Y-axis. Furthermore, considering the impracticality and extensive duration required to run the entire large dataset on a CPU-only setup, we chose to extrapolate CPU-only runtimes from a smaller sampled subset rather than conducting a direct measurement, in order to conserve time and computing resources.
Analyzing the results depicted in the figure, the disparity in running time is striking, spanning four orders of magnitude. For instance, screening about 30,000 decoys for a protein similar to ACETYLCHOLINESTERASE, which has approximately 8,000 atoms, takes only 5 seconds using our GPU-accelerated pipeline, achieving a throughput of 5455 decoys per second. In contrast, the same task requires 6781 seconds (nearly 2 hours) when using only the CPU with 18 cores, managing to process only 4 decoys per second. Notably, these results do not even account for the use of our Ligand Vibration Technique.

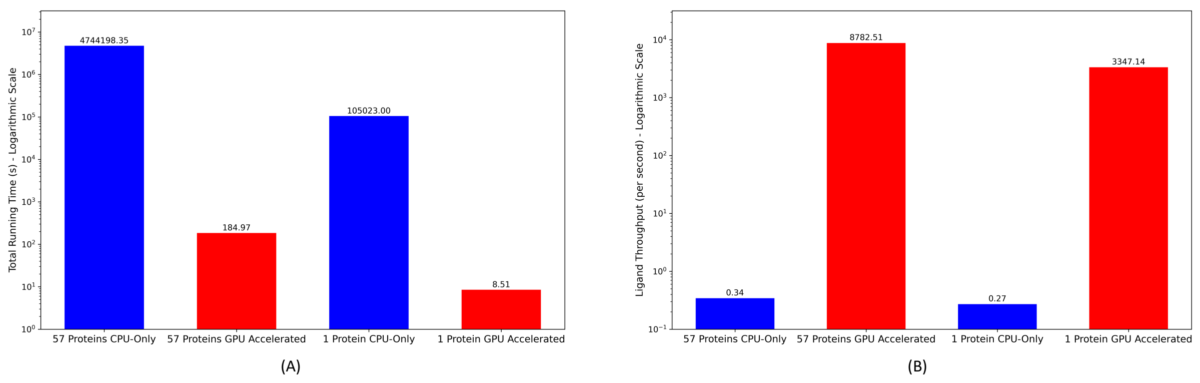


Figure 4.23: Comparative Analysis of our HPC Pipeline Performance with Ligand Vibration

The implementation of the Ligand Vibration Technique further accentuates the discrepancy in performance, as shown in Figure 4.23, where the total running time differs by four orders of magnitude. As depicted in Panel A, for a single protein with Vibration activated in 27 directions, the screening of its 28,500 decoys requires 105,023 seconds, approximately 27 hours. Remarkably, conducting the entire CASF-2016 Screening Power test, which involves 1,624,500 decoys for all 57 test proteins, takes nearly 55 days on a CPU-only setup. However, the efficiency significantly improves with our GPU acceleration. Screening 28,500 decoys for a single protein now takes just 8.51 seconds, while completing the entire Screening Power test requires only 184 seconds. The throughput

astonishingly reaches approximately 3,300 and 8,700 ligands per second, respectively. It is noteworthy that while the throughput with CPU-only shows little variation between different scenarios, it differs notably post-GPU acceleration. This discrepancy is attributed not only to the significant variation in the number of atoms among different proteins but also to the startup time involved in GPU acceleration. Such non-computational overhead includes tasks like memory allocation, freeing up GPU memory, and memory transfer.
Additionally, a critical observation from Figures 4.22 and 4.23 is the marked contrast in CPU-only runtimes between scenarios with and without the Ligand Vibration Technique, which shows an approximately 15-fold difference. In contrast, the disparity in total runtime post-GPU acceleration between using and not using the Ligand Vibration Technique is only about 20 seconds, essentially negligible. This fact further highlights the effectiveness of the optimizations applied to the Ligand Vibration Technique, as discussed in Section 3.5.3. Considering its significant contribution to prediction accuracy, the activation of the Ligand Vibration component in practical applications is highly advantageous.

5 | Conclusions and Future Directions

In the drug discovery cycle, the persistent challenges of extended timelines, substantial costs, and high failure rates continue to be critical issues addressed by both academia and industry. With the advent of advanced in-silico methods, there has been an alleviation of these challenges.

In this thesis, we introduced a novel contribution to the field of in-silico drug discovery, particularly in the Lead Identification phase. Our focus is predominantly on Structure-Based Virtual Screening (SBVS). This process entails the sampling of conformations for each Protein-Ligand Complex during Molecular docking, followed by the evaluation of these conformations using a Scoring function. The ultimate goal is to identify a set of optimal ligands for the target protein and advance them to wet laboratories for further experimental validation. To achieve this, we developed 'DrugXGBScore', a hybrid scoring function designed to maximize computational performance while maintaining high prediction accuracy to meet the High-Throughput Virtual Screening (HTVS) requirement. This Hybrid Scoring Function is a linear combination of Knowledge-based and machine-learning scoring functions. It harnesses the straightforward structure of the Knowledge-based scoring function for seamless integration with our high-performance computing pipeline. Concurrently, the machine learning scoring function significantly enhances prediction accuracy. Following this, we deployed the entire Hybrid Scoring Function into an HPC pipeline, tailor-made for our DrugXGBScore. Leveraging the advanced capabilities of our state-of-the-art Nvidia A-100 high-performance GPU, this HPC pipeline ensures that the computational demands of screening suitable ligands from extensive drug molecule libraries are adequately met.

Ultimately, our DrugXGBScore underwent comprehensive evaluation, rigorously assessing both its prediction accuracy and computational performance. In terms of prediction accuracy, our analysis of the CASF-2016 Power test results revealed that DrugXGBScore's Scoring and Ranking Power nearly matched the best levels among the benchmark scoring functions. Furthermore, it also achieved a mid-upper tier performance in the Docking and Screening Power tests. This performance of DrugXGBScore effectively meets our established criteria for prediction accuracy. Simultaneously, in terms of computing performance, our HPC pipeline demonstrated a remarkable improvement. When screening 28,500 decoys for the target protein ACETYLCHOLINESTERASE, the total execution time using our HPC pipeline was over four orders of magnitude faster compared to relying solely on the CPU. Specifically, this task required only 8.5 seconds with our HPC pipeline, in sharp contrast to nearly 27 hours with just the CPU. Additionally, the throughput increased dramatically from an average of 0.27 decoys per second to 3,347 decoys per second. Notably, this throughput is expected to rise further with larger volumes of decoys, as they more effectively distribute the startup time. Overall, the large increase in computational performance with our HPC pipeline, as opposed to using only the CPU, is extraordinary. However, our research is not without limitations. A primary challenge we encountered was the quality of the training datasets, which significantly impacted our model training process. It is clear that the dataset quality had a considerable effect on the final prediction accuracy. Improving it could substantially enhance prediction accuracy, thus increasing the success probability of the ligands identified for further testing in wet laboratory experiments. Crucially, such improvements in dataset quality would not adversely affect computing performance. With access to higher-quality training datasets, our DrugXGBScore could be significantly improved from its current state.

Looking toward future directions, we are confident that the research presented in this thesis holds substantial scalability and possibly offers meaningful guidance for advancing the field of Structure-Based Virtual Screening (SBVS). Firstly, addressing dataset quality issues is crucial. As discussed in the previous paragraph, high-quality datasets can significantly improve prediction accuracy without adversely affecting computing performance. Given the high costs associated with data collection, in scenarios where high-quality protein-ligand structure or binding affinity data are not readily available, we could explore the use of emerging technologies, such as DeepMind's AlphaFold [35], to optimize our existing datasets. Secondly, regarding the scoring function, the application of machine learning algorithms has already yielded substantial improvements in prediction accuracy. Our future efforts should not be limited to the XGBoost algorithm currently in use. We could experiment with various machine/deep learning algorithms, even popular Large Language Models (LLMs) [3], to further enhance prediction accuracy. Lastly, in terms of computing performance, we could utilize high-performance GPU clusters or even large-scale computing infrastructures. This approach would significantly enhance the computational capabilities of our HPC pipeline, potentially transforming both the running time and throughput to an incredible higher level.

Bibliography

[1] Openmp. `https://www.openmp.org/`. Accessed: 2023.

[2] N. Adiga, G. Almasi, G. Almasi, Y. Aridor, R. Barik, D. Beece, R. Bellofatto, G. Bhanot, R. Bickford, M. Blumrich, A. Bright, J. Brunheroto, C. Cascaval, J. Castanos, W. Chan, L. Ceze, P. Coteus, S. Chatterjee, D. Chen, G. Chiu, T. Cipolla, P. Crumley, K. Desai, A. Deutsch, T. Domany, M. Dombrowa, W. Donath, M. Eleftheriou, C. Erway, J. Esch, B. Fitch, J. Gagliano, A. Gara, R. Garg, R. Germain, M. Giampapa, B. Gopalsamy, J. Gunnels, M. Gupta, F. Gustavson, S. Hall, R. Haring, D. Heidel, P. Heidelberger, L. Herger, D. Hoenicke, R. Jackson, T. Jamal-Eddine, G. Kopcsay, E. Krevat, M. Kurhekar, A. Lanzetta, D. Lieber, L. Liu, M. Lu, M. Mendell, A. Misra, Y. Moatti, L. Mok, J. Moreira, B. Nathanson, M. Newton, M. Ohmacht, A. Oliner, V. Pandit, R. Pudota, R. Rand, R. Regan, B. Rubin, A. Ruehli, S. Rus, R. Sahoo, A. Sanomiya, E. Schenfeld, M. Sharma, E. Shmueli, S. Singh, P. Song, V. Srinivasan, B. Steinmacher-Burow, K. Strauss, C. Surovic, R. Swetz, T. Takken, R. Tremaine, M. Tsao, A. Umamaheshwaran, P. Verma, P. Vranas, T. Ward, M. Wazlowski, W. Barrett, C. Engel, B. Drehmel, B. Hilgart, D. Hill, F. Kasemkhani, D. Krolak, C. Li, T. Liebsch, J. Marcella, A. Muff, A. Okomo, M. Rouse, A. Schram, M. Tubbs, G. Ulsh, C. Wait, J. Wittrup, M. Bae, K. Dockser, L. Kissel, M. Seager, J. Vetter, and K. Yates. An overview of the Blue-Gene/l supercomputer. In *SC '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 60–60. doi: 10.1109/SC.2002.10017. ISSN: 1063-9535.

[3] M. R. AI4Science and M. A. Quantum. The impact of large language models on scientific discovery: a preliminary study using GPT-4. URL `http://arxiv.org/abs/2311.07361`.

[4] W. J. Allen and R. C. Rizzo. Implementation of the hungarian algorithm to account for ligand symmetry and similarity in structure-based design. 54(2):518–529. ISSN 1549-9596. doi: 10.1021/ci400534h. URL `https://doi.org/10.1021/ci400534h`. Publisher: American Chemical Society.

[5] J. Audie and S. Scarlata. A novel empirical free energy function that explains

and predicts protein–protein binding affinities. 129(2):198–211. ISSN 0301-4622. doi: 10.1016/j.bpc.2007.05.021. URL https://www.sciencedirect.com/science/article/pii/S0301462207001536.

[6] M. Awale, R. van Deursen, and J.-L. Reymond. MQN-mapplet: Visualization of chemical space with interactive maps of DrugBank, ChEMBL, PubChem, GDB-11, and GDB-13. 53(2):509–518. ISSN 1549-9596. doi: 10.1021/ci300513m. URL https://doi.org/10.1021/ci300513m. Publisher: American Chemical Society.

[7] P. Badrinarayan and G. N. Sastry. Virtual high throughput screening in new lead identification. 14(10):840–860, 2011. ISSN 1875-5402. doi: 10.2174/138620711797537102.

[8] I. Bahar and R. L. Jernigan. Inter-residue potentials in globular proteins and the dominance of highly specific hydrophilic interactions at close separation 1 1 edited by b. honig. 266(1):195–214. ISSN 0022-2836. doi: 10.1006/jmbi.1996.0758. URL https://www.sciencedirect.com/science/article/pii/S0022283696907585.

[9] M. H. Baig, K. Ahmad, M. Adil, Z. A. Khan, M. I. Khan, M. Lohani, M. S. Khan, and M. A. Kamal. Drug discovery and in silico techniques: A mini-review. 04(1), 2014. ISSN 23296674. doi: 10.4172/2329-6674.1000123. URL http://www.omicsgroup.org/journals/drug-discovery-and-in-silico-techniques-a-minireview-2329-6674-1000123.php?aid=43621.

[10] H.-J. Bohm. The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure. 8(3):243–256. ISSN 0920-654X, 1573-4951. doi: 10.1007/BF00126743. URL http://link.springer.com/10.1007/BF00126743.

[11] N. Brooijmans and I. D. Kuntz. Molecular recognition and docking algorithms. 32(1):335–373. doi: 10.1146/annurev.biophys.32.110601.142532. URL https://doi.org/10.1146/annurev.biophys.32.110601.142532. _eprint: https://doi.org/10.1146/annurev.biophys.32.110601.142532.

[12] P. Canada. Glossary, 2023. URL https://pharmacologycanada.org/Glossary. Accessed: 6 Oct 2023.

[13] C. Catana and P. F. W. Stouten. Novel, customizable scoring functions, parameterized using n-PLS, for structure-based drug discovery. 47(1):85–91. ISSN 1549-9596. doi: 10.1021/ci600357t. URL https://doi.org/10.1021/ci600357t. Publisher: American Chemical Society.

[14] P. S. Charifson, J. J. Corkery, M. A. Murcko, and W. P. Walters. Consensus scoring: A method for obtaining improved hit rates from docking databases of three-dimensional structures into proteins. 42(25):5100–5109. ISSN 0022-2623. doi: 10.1021/jm990352k. URL `https://doi.org/10.1021/jm990352k`. Publisher: American Chemical Society.

[15] P. Chaskar, V. Zoete, and U. F. Röhrig. Toward on-the-fly quantum mechanical/molecular mechanical (QM/MM) docking: Development and benchmark of a scoring function. 54(11):3137–3152. ISSN 1549-9596. doi: 10.1021/ci5004152. URL `https://doi.org/10.1021/ci5004152`. Publisher: American Chemical Society.

[16] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. doi: 10.1145/2939672.2939785. URL `http://arxiv.org/abs/1603.02754`.

[17] Cineca. Leonardo supercomputer - hpc system, 2023. URL `https://leonardo-supercomputer.cineca.eu/hpc-system/`. Accessed: 2 Sep 2023.

[18] N. Corporation. Cuda c programming guide. Available at: `https://docs.nvidia.com/cuda/cuda-c-programming-guide/`, 2023. Accessed: 2023-09-05.

[19] X. Developers. Xgboost parameters documentation. URL `https://xgboost.readthedocs.io/en/stable/parameter.html`. Accessed: 2023.

[20] J. A. DiMasi, R. W. Hansen, and H. G. Grabowski. The price of innovation: new estimates of drug development costs. 22(2):151–185, 2003. ISSN 0167-6296. doi: 10.1016/S0167-6296(02)00126-1. URL `https://www.sciencedirect.com/science/article/pii/S0167629602001261`.

[21] J. Dittrich, D. Schmidt, C. Pfleger, and H. Gohlke. Converging a knowledge-based scoring function: DrugScore ^{2018}. 59(1):509–521. ISSN 1549-9596, 1549-960X. doi: 10.1021/acs.jcim.8b00582. URL `https://pubs.acs.org/doi/10.1021/acs.jcim.8b00582`.

[22] M. D. Eldridge, C. W. Murray, T. R. Auton, G. V. Paolini, and R. P. Mee. Empirical scoring functions: I. the development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. 11(5):425–445. ISSN 1573-4951. doi: 10.1023/A:1007996124545. URL `https://doi.org/10.1023/A:1007996124545`.

[23] J. Fan, A. Fu, and L. Zhang. Progress in molecular docking. 7(2):83–89. ISSN

2095-4697. doi: 10.1007/s40484-019-0172-y. URL https://doi.org/10.1007/s40484-019-0172-y.

[24] J. H. Friedman. Greedy function approximation: A gradient boosting machine. 29 (5):1189–1232. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1013203451. URL https://projecteuclid.org/journals/annals-of-statistics/volume-29/issue-5/Greedy-function-approximation-A-gradient-boosting-machine/10.1214/aos/1013203451.full. Publisher: Institute of Mathematical Statistics.

[25] R. A. Friesner, J. L. Banks, R. B. Murphy, T. A. Halgren, J. J. Klicic, D. T. Mainz, M. P. Repasky, E. H. Knoll, M. Shelley, J. K. Perry, D. E. Shaw, P. Francis, and P. S. Shenkin. Glide: A new approach for rapid, accurate docking and scoring. 1. method and assessment of docking accuracy. 47(7):1739–1749. ISSN 0022-2623. doi: 10.1021/jm0306430. URL https://doi.org/10.1021/jm0306430. Publisher: American Chemical Society.

[26] D. Gadioli, E. Vitali, F. Ficarelli, C. Latini, C. Manelfi, C. Talarico, C. Silvano, C. Cavazzoni, G. Palermo, and A. R. Beccari. EXSCALATE: An extreme-scale virtual screening platform for drug discovery targeting polypharmacology to fight SARS-CoV-2. 11(1):170–181. ISSN 2168-6750. doi: 10.1109/TETC.2022.3187134. Conference Name: IEEE Transactions on Emerging Topics in Computing.

[27] C. Garcia-Hernandez, A. Fernández, and F. Serratosa. Ligand-based virtual screening using graph edit distance as molecular similarity measure. 59(4):1410–1421. ISSN 1549-9596. doi: 10.1021/acs.jcim.8b00820. URL https://doi.org/10.1021/acs.jcim.8b00820. Publisher: American Chemical Society.

[28] H. Gohlke, M. Hendlich, and G. Klebe. Knowledge-based scoring function to predict protein-ligand interactions. 295(2):337–356. ISSN 0022-2836. doi: 10.1006/jmbi.1999.3371. URL https://www.sciencedirect.com/science/article/pii/S0022283699933715.

[29] M. Hendlich. Databases for protein–ligand complexes. 54(6): 1178–1182. ISSN 0907-4449. doi: 10.1107/S0907444998007124. URL //scripts.iucr.org/cgi-bin/paper?ba0006. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1107/S0907444998007124.

[30] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 5th edition, 2012. ISBN 978-0-12-383872-8.

[31] S.-Y. Huang, S. Z. Grinter, and X. Zou. Scoring functions and their evaluation methods for protein–ligand docking: recent advances and future directions. 12

(40):12899–12908. ISSN 1463-9084. doi: 10.1039/C0CP00151A. URL https://pubs.rsc.org/en/content/articlelanding/2010/cp/c0cp00151a. Publisher: The Royal Society of Chemistry.

[32] A. Inc. Metal for developers, 2023. Available at: https://developer.apple.com/metal/. Accessed: 2023-09-05.

[33] Intel. oneapi: What is it?, 2022. Available at: https://www.intel.com/content/www/us/en/developer/articles/technical/oneapi-what-is-it.html#gs.5bcnsj. Accessed: 2023-09-05.

[34] M. Johnson and G. Maggiora. *Concepts and Applications of Molecular Similarity*. John Wiley, 1990. ISBN 978-0-471-62175-1.

[35] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. 596(7873):583–589. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2. URL https://www.nature.com/articles/s41586-021-03819-2. Number: 7873 Publisher: Nature Publishing Group.

[36] M. Kadukova and S. Grudinin. Convex-PL: a novel knowledge-based potential for protein-ligand interactions deduced from structural databases using convex optimization. 31(10):943–958. ISSN 1573-4951. doi: 10.1007/s10822-017-0068-8. URL https://doi.org/10.1007/s10822-017-0068-8.

[37] I. Khanna. Drug discovery in pharmaceutical industry: productivity challenges and trends. 17(19):1088–1102, 2012. ISSN 1359-6446. doi: 10.1016/j.drudis.2012.05.007. URL https://www.sciencedirect.com/science/article/pii/S1359644612001833.

[38] D. B. Kitchen, H. Decornez, J. R. Furr, and J. Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. 3(11):935–949. ISSN 1474-1784. doi: 10.1038/nrd1549. URL https://www.nature.com/articles/nrd1549. Number: 11 Publisher: Nature Publishing Group.

[39] O. Korb, T. Stützle, and T. E. Exner. Empirical scoring functions for advanced protein-ligand docking with PLANTS. 49(1):84–96. ISSN 1549-9596. doi:

10.1021/ci800298z. URL https://doi.org/10.1021/ci800298z. Publisher: American Chemical Society.

[40] D. E. Koshland. The key-lock theory and the induced fit theory. 33(2324):2375–2378. ISSN 0570-0833, 1521-3773. doi: 10.1002/anie.199423751. URL https://onlinelibrary.wiley.com/doi/10.1002/anie.199423751.

[41] A. A. Kossiakoff, M. Randal, J. Guenot, and C. Eignebrot. Variability of conformations at crystal contacts in BPTI represent true low-energy structures: Correspondence among lattice packing and molecular dynamics structures. 14(1):65–74. ISSN 1097-0134. doi: 10.1002/prot.340140108. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.340140108. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/prot.340140108.

[42] J. Li, A. Fu, and L. Zhang. An overview of scoring functions used for protein–ligand interactions in molecular docking. 11(2):320–328, 2019. ISSN 1867-1462. doi: 10.1007/s12539-019-00327-w. URL https://doi.org/10.1007/s12539-019-00327-w.

[43] J. Lu, X. Hou, C. Wang, and Y. Zhang. Incorporating explicit water molecules and ligand conformation stability in machine-learning scoring functions. 59(11):4540–4549. ISSN 1549-9596. doi: 10.1021/acs.jcim.9b00645. URL https://doi.org/10.1021/acs.jcim.9b00645. Publisher: American Chemical Society.

[44] P. D. Lyne. Structure-based virtual screening: an overview. 7(20):1047–1055. ISSN 1359-6446. doi: 10.1016/S1359-6446(02)02483-2. URL https://www.sciencedirect.com/science/article/pii/S1359644602024832.

[45] L. M. Mayr and D. Bojanic. Novel trends in high-throughput screening. 9(5):580–588, 2009. ISSN 1471-4892. doi: 10.1016/j.coph.2009.08.004. URL https://www.sciencedirect.com/science/article/pii/S1471489209001283.

[46] E. C. Meng, B. K. Shoichet, and I. D. Kuntz. Automated docking with grid-based energy evaluation. 13(4):505–524. ISSN 1096-987X. doi: 10.1002/jcc.540130412. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.540130412. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.540130412.

[47] H. L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. 5(2):107–113. ISSN 0021-9576. doi: 10.1021/c160017a018. URL https://doi.org/10.1021/c160017a018. Publisher: American Chemical Society.

[48] J. L. Morrison, R. Breitling, D. J. Higham, and D. R. Gilbert. A lock-and-key model

for protein–protein interactions. 22(16):2012–2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl338. URL `https://doi.org/10.1093/bioinformatics/btl338`.

[49] G. Neudert and G. Klebe. fconv: format conversion, manipulation and feature computation of molecular data. 27(7):1021–1022. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr055. URL `https://doi.org/10.1093/bioinformatics/btr055`.

[50] NVIDIA Corporation. Nvidia a100 gpu. `https://www.nvidia.com/en-us/data-center/a100/`. Accessed: 2023.

[51] N. I. of Health (NIH). The sybyl data structure document. `https://tccc.iesl.forth.gr/education/local/quantum/molecular_modeling/guide_documents/SYBYL_data_document.html`, 1994. Accessed: 2023.

[52] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design - The Hardware/Software Interface*. Morgan Kaufmann, 5th edition, 2014. ISBN 978-0-12-407726-3.

[53] PDBbind. Pdbbind database, 2023. URL `http://www.pdbbind.org.cn/`. Accessed: 3 Oct 2023.

[54] D. A. Pearlman and P. S. Charifson. Are free energy calculations useful in practice? a comparison with rapid scoring functions for the p38 MAP kinase protein system. 44(21):3417–3423. ISSN 0022-2623. doi: 10.1021/jm0100279. URL `https://doi.org/10.1021/jm0100279`. Publisher: American Chemical Society.

[55] S. Pushpakom, F. Iorio, P. A. Eyers, K. J. Escott, S. Hopper, A. Wells, A. Doig, T. Guilliams, J. Latimer, C. McNamee, A. Norris, P. Sanseau, D. Cavalla, and M. Pirmohamed. Drug repurposing: progress, challenges and recommendations. 18(1):41–58. ISSN 1474-1784. doi: 10.1038/nrd.2018.168. URL `https://www.nature.com/articles/nrd.2018.168`. Number: 1 Publisher: Nature Publishing Group.

[56] G. Rastelli and L. Pinzi. Refinement and rescoring of virtual screening results. 7. ISSN 2296-2646. URL `https://www.frontiersin.org/articles/10.3389/fchem.2019.00498`.

[57] D. Rogers and M. Hahn. Extended-connectivity fingerprints. 50(5):742–754. ISSN 1549-9596. doi: 10.1021/ci100050t. URL `https://doi.org/10.1021/ci100050t`. Publisher: American Chemical Society.

[58] G. Schneider and U. Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663, 2005. doi: 10.1038/nrd1799. URL `https://doi.org/10.1038/nrd1799`.

[59] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of bayesian optimization. 104(1):148–175. ISSN 1558-2256. doi: 10.1109/JPROC.2015.2494218. URL `https://ieeexplore.ieee.org/document/7352306?denied=`. Conference Name: Proceedings of the IEEE.

[60] B. K. Shoichet. Virtual screening of chemical libraries. 432(7019):862–865, 2004. ISSN 1476-4687. doi: 10.1038/nature03197. URL `https://www.nature.com/articles/nature03197`. Number: 7019 Publisher: Nature Publishing Group.

[61] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 3 edition, 2012. ISBN 978-1-133-18779-0. ISBN-10: 1-133-18779-X.

[62] N. Stiefl, I. A. Watson, K. Baumann, and A. Zaliani. ErG: 2d pharmacophore descriptions for scaffold hopping. 46(1):208–220. ISSN 1549-9596. doi: 10.1021/ci050457y. URL `https://doi.org/10.1021/ci050457y`. Publisher: American Chemical Society.

[63] M. Su, Q. Yang, Y. Du, G. Feng, Z. Liu, Y. Li, and R. Wang. Comparative assessment of scoring functions: The CASF-2016 update. 59(2):895–913. ISSN 1549-9596. doi: 10.1021/acs.jcim.8b00545. URL `https://doi.org/10.1021/acs.jcim.8b00545`. Publisher: American Chemical Society.

[64] O. B. D. Team. Open babel: The open source chemistry toolbox, 2023. URL `http://openbabel.org`. Accessed: 2023.

[65] G. C. Terstappen and A. Reggiani. In silico research in drug discovery. 22(1):23–26. ISSN 0165-6147. doi: 10.1016/S0165-6147(00)01584-4. URL `https://www.sciencedirect.com/science/article/pii/S0165614700015844`.

[66] Top500. List of top supercomputers, 2023. URL `https://www.top500.org/`. Data from the June 2023 ranking, 61st edition of the TOP500.

[67] H. F. G. Velec, H. Gohlke, and G. Klebe. DrugScoreCSDKnowledge-based scoring function derived from small molecule crystal data with superior recognition rate of near-native ligand poses and better affinity prediction. 48(20):6296–6303. ISSN 0022-2623. doi: 10.1021/jm050436v. URL `https://doi.org/10.1021/jm050436v`. Publisher: American Chemical Society.

[68] C. Wang and Y. Zhang. Improving scoring-docking-screening powers of protein–ligand scoring functions using random forest. 38(3): 169–177, 2017. ISSN 1096-987X. doi: 10.1002/jcc.24667. URL

https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.24667. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.24667.

[69] R. Wang, L. Lai, and S. Wang. Further development and validation of empirical scoring functions for structure-based binding affinity prediction. 16(1):11–26. ISSN 1573-4951. doi: 10.1023/A:1016357811882. URL https://doi.org/10.1023/A:1016357811882.

[70] S. J. Weiner, P. A. Kollman, D. A. Case, U. C. Singh, C. Ghio, G. Alagona, S. Profeta, and P. Weiner. A new force field for molecular mechanical simulation of nucleic acids and proteins. 106(3):765–784, . ISSN 0002-7863. doi: 10.1021/ja00315a051. URL https://doi.org/10.1021/ja00315a051. Publisher: American Chemical Society.

[71] S. J. Weiner, P. A. Kollman, D. T. Nguyen, and D. A. Case. An all atom force field for simulations of proteins and nucleic acids. 7 (2):230–252, . ISSN 1096-987X. doi: 10.1002/jcc.540070216. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.540070216. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.540070216.

[72] P. Willett. Similarity-based virtual screening using 2d fingerprints. 11(23):1046–1053. ISSN 1359-6446. doi: 10.1016/j.drudis.2006.10.005. URL https://www.sciencedirect.com/science/article/pii/S1359644606004193.

[73] L. Xue and J. Bajorath. Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening. 3(5):363–372. ISSN 13862073. doi: 10.2174/1386207003331454. URL http://www.eurekaselect.com/openurl/content.php?genre=article&issn=1386-2073&volume=3&issue=5&spage=363.

[74] L. Zhang, H.-X. Ai, S.-M. Li, M.-Y. Qi, J. Zhao, Q. Zhao, and H.-S. Liu. Virtual screening approach to identifying influenza virus neuraminidase inhibitors using molecular docking combined with machine-learning-based scoring function. 8 (47):83142–83154. ISSN 1949-2553. doi: 10.18632/oncotarget.20915. URL https://www.oncotarget.com/article/20915/text/. Publisher: Impact Journals.

List of Figures

List of Tables

Acknowledgements

I would like to express my deepest gratitude to my advisors, Prof. Gianluca Palermo, Davide Gadioli, and Gianmarco Accordi, for their invaluable guidance, support, and expertise throughout my Master's thesis journey.