UCAM

UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

Programa de Doctorado en Tecnologías de la Computación e Ingeniería Ambiental

Enhancing Molecular Docking with Deep Q-Networks

Autor:
D. Antonio Serrano Fernández

Directores:
Dr. D. Andrés Bueno Crespo
Dr. D. José Luis Abellán Miguel

Murcia, septiembre de 2021

UCAM

UNIVERSIDAD CATÓLICA
DE MURCIA

ESCUELA INTERNACIONAL DE DOCTORADO

Programa de Doctorado en Tecnologías de la Computación e Ingeniería Ambiental

Enhancing Molecular Docking with Deep Q-Networks

Autor:
D. Antonio Serrano Fernández

Directores:
Dr. D. Andrés Bueno Crespo
Dr. D. José Luis Abellán Miguel

Murcia, septiembre de 2021

EIDUCAM
Escuela Internacional
de Doctorado

UCAM
UNIVERSIDAD CATÓLICA DE MURCIA

AUTORIZACIÓN DEL DIRECTOR DE LA TESIS PARA SU PRESENTACIÓN

El Dr. D. Andrés Bueno Crespo y el Dr. D. José Luis Abellán Miguel como Directores[1] de la Tesis Doctoral titulada "Enhancing Molecular Docking with Deep Q-Networks" realizada por D. Antonio Serrano Fernández en el Programa de Doctorado en Tecnologías de la Computación e Ingeniería Ambiental, **autoriza su presentación a trámite** dado que reúne las condiciones necesarias para su defensa.

Lo que firmo, para dar cumplimiento al Real Decreto 99/2011 de 28 de enero, en Murcia a 28 de septiembre de 2021.

Fdo: Andrés Bueno Crespo

Fdo: José Luis Abellán Miguel

[1] Si la Tesis está dirigida por más de un Director tienen que constar y firmar ambos.

A Belén, la mujer de mi vida y la persona más fiel que he conocido nunca. Por estar siempre ahí, en las alegrías y dificultades de este largo camino para cumplir mi sueño.

A mis padres, quienes, con su amor y generosidad, siempre me han guiado en el discurrir del tren de la vida. Sin ellos no hubiera sido posible llegar hasta esta estación de convertirme en doctor.

A mi hermano y mi cuñada Pilar, mi faro de referencia y apoyo incondicional para todo.

A mi Tita Choni, siempre atenta y diligente para echarme una mano en lo que haga falta.

A mis suegros y a la familia Escobar-García, mi segunda familia, por su apoyo y su cariño brindado.

Agradecimientos

Gracias a Dios por bendecirme con la vida y su buena estrella, la cual me guiado en cada paso que he ido dando hasta ahora.

Gracias a mis directores de tesis por aconsejarme durante todo el camino, especialmente cuando éste se ponía cuesta arriba, me flaqueaban los ánimos y se me nublaba la razón. A José Luis por resolverme tropecientas dudas cuando más perdido estaba. A Andrés por resolverme otras tantas y ayudarme a desconectar con sus chascarrillos.

Mi agradecimiento también para Belén, como directora del Grado en Ingeniería Informática, por acogerme y permitirme continuar el camino que me había propuesto. También por facilitar la coordinación de mi formación del doctorado con mis labores docentes y por enseñarme unos cuantos trucos de la vieja escuela a la hora de impartir clases.

Gracias a Estrella, vicerrectora de Investigación, sin su cariño, respaldo y confianza, sencillamente, no habría podido llegar hasta aquí.

Mi agradecimiento con afecto para Baldo, siempre solícito y generoso para prestar ayuda a los demás de buena gana. Sin él y su METADOCK, mi trabajo de la tesis doctoral sencillamente no luciría igual.

Gracias a Horacio, por compartir su sabiduría y conocimiento en los trabajos de investigación que hemos desarrollado juntos, así como aquellos recursos computacionales que he necesitado por su parte para la elaboración de esta tesis. Gracias igualmente por estar siempre dispuesto a responder largos emails repletos de dudas sobre Docking y a lo que hiciese falta.

Mis agradecimientos también para Chema, por compartir servidores asociados a los proyectos en los que figuraba IP, así como trucos para lidiar con los revisores más duros de la revista ASOC. Igualmente, gracias por ayudar a financiar el primer congreso de mi

carrera investigadora en el ICPP'18 de Eugene, Oregón. Nunca olvidaré ese viaje.

A todos los compañeros del departamento. Es difícil encontrar un oasis donde se respire tanto cariño y compañerismo. Luz, Onia, Miguel Ángel, Belén, Jesús, Antonio Llanes, Paco, Mada, Fernando, Baldo, Manu, Antonio Jesús, Juan, Pedro, Joaquín y Javier. También a los que ya no están, pero se les echa en falta: Raquel, Andrés Muñoz y Chema.

My heartfelt thanks also go to Dr. Benjamin Schubert and every member of the Translational Immunoinformatics research group at Institute of Computational Biology - Helmholtz Zentrum München in Munich. Although we couldn't meet in person due to the pandemic, I had a lot of fun playing that virtual escape room and in the poster session at the Group Day. Those memories will last forever.

Por último, muchas gracias a la Universidad Católica de Murcia (UCAM) en su conjunto por la confianza depositada en mi y, en especial, a su presidente D. José Luis Mendoza Pérez. Sin su acogida y apoyo financiero esta tesis no habría sido posible.

Enhancing Molecular Docking with Deep Q-Networks

RESUMEN

El descubrimiento de fármacos es un proceso largo y costoso que suele durar entre 10 y 15 años, desde la evaluación inicial de candidatos farmacológicos hasta la aprobación final por parte de los organismos reguladores correspondientes. Por este motivo, simulaciones moleculares por computador, conocidas como Virtual Screening (VS) (o Cribado Virtual), se utilizan a menudo para predecir los candidatos a fármacos durante las primeras etapas de su desarrollo. Uno de los métodos más utilizados en el VS es el llamado Docking Molecular, o simplemente abreviado como Docking (en español, Acoplamiento Molecular). El objetivo de este método es resolver el problema de las Interacciones Proteína-Ligando (PLDP) o Docking. Dicho de otro modo, se trata de predecir las conformaciones 3D en las que un candidato farmacológico (también conocido como ligando) se acopla a un receptor determinado (normalmente una proteína) en un punto concreto de su superficie. Los métodos tradicionales de Docking se basan en procedimientos de optimización de funciones de puntuación (o de scoring) siguiendo determinadas heurísticas. Se trata de funciones matemáticas que modelan las interacciones moleculares.

Estos métodos se caracterizan por ser computacionalmente costosos. De esta manera, en esta tesis se pretende aprovechar los prometedores algoritmos de Deep RL para mejorar la resolución del problema de Docking. Para ello, el hilo conductor de esta tesis doctoral son las diferentes alternativas de representación de las moléculas de la escena de Docking que serán utilizadas como datos de entrada de dichos algoritmos.

En consecuencia, primero se replantea el problema PLDP como uno de Aprendizaje por Refuerzo (RL). Acto seguido, se construye un sistema básico basado en el algoritmo

de Deep Q-Network (DQN), originalmente diseñado para enseñar a agentes artificiales a jugar a videojuegos de la consola de Atari 2600. En segundo lugar, se utiliza una implementación, denominada QN-Docking, basada en un vector de características sencillo para la representación molecular. Dicha implementación es testada en un entorno con un receptor relativamente pequeño y un espacio de acciones limitado. Los resultados de la fase de predicción muestran que QN-Docking consigue un aumento de velocidad 8 veces mayor en comparación con métodos estocásticos como METADOCK 2. Dicho programa es un nuevo software de alto rendimiento que incluye diversas metaheurísticas para el Acoplamiento Molecular. Por último, una implementación alternativa basada en imágenes, MVDQN, es testada en el mismo escenario que QN-Docking. Los resultados muestran un rendimiento similar al de la primer implementación durante la fase de entrenamiento. Sin embargo, en la fase de predicción los resultados son mixtos. El agente actúa de forma subóptima en varias de las posiciones de partida establecidas en el experimento. Este escenario final parece prometedor, no obstante, ya que hay mucho margen de mejora para seguir puliendo el algoritmo y mejorar la representación molecular.

En resumen, estos resultados suponen un valioso hito en el desarrollo de un método basado en Inteligencia Artificial más rápido y efectivo para resolver el problema PLDP en comparación con métodos más tradicionales.

Palabras clave: Informática, Inteligencia Artificial, Redes Neuronales, Farmacología molecular.

Enhancing Molecular Docking with Deep Q-Networks.

ABSTRACT

Drug discovery is a long and expensive process that normally takes 10-15 years from primary evaluation to regulator's approval. As a result, molecular computer simulations known as Virtual Screening (VS) are often used to predict drug candidates during the first stages. One of the most widely used methods in VS is Molecular Docking—or just Docking. The goal of this method is to solve the Protein-Ligand Docking Prediction (PLDP) problem. In other words, to predict the 3D conformations where a pharmacological candidate (also known as the ligand) binds to a given receptor (normally a protein) in a particular spot around its surface. Traditional Docking methods are based on optimization procedures of scoring functions following specific heuristics. These mathematical functions model the molecular interactions.

Such methods are characterized by being computationally expensive. Thus, in this dissertation it is intended to take advantage of the promising algorithms of Deep RL to enhance the resolution of the Docking problem. To do so, the common thread of this doctoral thesis is the different alternatives of representing molecules from the Docking scene to be used as input for those algorithms.

Consequently, it is first reframed the PLDP problem as one of Reinforcement Learning (RL). Then, it is built a basic system based on the algorithm of Deep Q-Network (DQN), originally designed to teach artificial agents to play Atari 2600's video games. Second, an implementation based on a simple feature vector for molecular representation called QN-Docking is used. This implementation is tested in an environment based on a relatively small receptor and limited action space. Results for the prediction phase show that QN-

Docking achieves 8 × speedup compared to stochastic methods such as METADOCK 2, a novel high-throughput parallel metaheuristic software for Docking. Finally, an alternative implementation based on images, MVDQN, is tested in the same setting. Results shows similar performance to QN-Docking during the training phase. However, the results are mixed in the prediction stage. The agent acts suboptimally in several of the starting positions set in the experiment. Nevertheless, this final scenario seems promising since there is much room for improvement to keep polishing the algorithm and improving the molecular representation.

In summary, these results entail a valuable milestone in developing a faster and effective Artificial-Intelligence-based method to solve the PLDP problem in comparison with more traditional methods.

Keywords: Computer Science, Artificial Intelligence, Neuronal Networks, Molecular Pharmacology.

Contents

List of Figures

Chapter 1

Introduction

1.1 Docking and Deep Reinforcement Learning

Drug development is known for being an excessively expensive, long, and difficult process. It often takes an average of 10-15 years from the initial steps to market launch [8, 46]. Drug discovery, in particular, includes the first steps of the entire drug development process. To this aim, thousands of pharmacological candidates—also know as ligands—are subsequently filtered in a workflow known as Virtual Screening (VS). VS is a computational method employed in drug discovery to seek libraries of ligands in order to identify those structures which are most likely to bind to a specific drug target, normally a protein or enzyme. They consist of molecular simulations performed in a fast and precise fashion to recreate the atomic interactions among molecules. A few hundred of the candidates are selected through VS previous to the High-Throughput Screening (HTS) phase. In that later phase, millions of pharmacological tests are conducted in order to check the validity of the potential drugs. Those HTS tests are performed with the assistance of robotic arms, data processing/control software, liquid handling devices, and sensitive detectors. Therefore, they are normally quite expensive compared to the in silico exper-

iments conducted with VS. Thus, VS methods have been extensively used to speed-up the first stages of drug development in the last decades.

One of the most effective methods in VS is Docking[114], applied to solve the problem known as Protein-Ligand Docking Prediction (PLDP). In 1982,Kuntz et al. [73] were the first to design and test a collection of algorithms to explore the geometrically viable alignments of a ligand and a target molecule. But it was not until the 1990s when it became widely used thanks to further improvements in the method itself [92], the significant improvement in raw computational power of computer systems, and the increment in available molecular structural data. Two notes must be made before continuing, for the sake of clarity and contextualization: (1) The term Molecular Docking, or just Docking, will be interchangeably used to refer to both the PLDP problem and the method to solve it from this point forward; and (2) The current dissertation will revolve around the interaction between a large target molecule (normally, a protein) and a small molecule (the ligand), not encompassing protein-protein Docking. Likewise, Docking consists of the exploration of ligand conformations adopted within the binding sites of macromolecular targets like proteins [33]. As a result, this method not only shortens the research-to-market cycles but also leads to enormous cost savings [63, 155]. However, it demands powerful high-performance computing platforms, advanced parallel programming models and complex algorithmic optimizations to deal with its intensive computational expense [28, 52].

In parallel, the field of Artificial Intelligence has gained a tremendous momentum in the last decade. This heyday is mostly due to the last achievements in the subfield of Machine Learning (ML), and in particular in Deep Learning (DL) [76]. DL is a family of algorithms based on learning data representations, whose most representative models are the Artificial Neural Networks (ANNs). These are vaguely inspired by the biological neural networks. They are made of many simple computing units (artificial neurons)

arranged in sequential layers [131]. If a given ANN has three or more layers, then it is described as a "deep" neural network. One of the strongest features of these powerful and flexible models is that they are able to automatically learn high-level abstractions of data. Consequently, DL has been successfully tested in a wide range of research and application fields such as image recognition, speech recognition, machine translation, self-driving cars, medical diagnosis, virtual personal assistants, stock market trading, online fraud detection, recommender systems, and power systems [29, 76, 135], to name just a few.

Furthermore, in the last lustrum there has been an increasing, vivid research trend of DL applied to drug discovery [12, 15, 45], specially since the astounding results achieved in Merck Kaggle [91] and NIH Tox21 [95] data challenges. Such trend has sharply increased in the last year because of the high interest in finding a cure, either a novel vaccine or a small pharmacological molecule, for the SARS-CoV-2 and stop the thousands of deaths that this coronavirus is causing worldwide [56, 74]. Recently, it has also achieved astonishing results in protein structure prediction [64, 136, 158]. The proposed system, named AlphaFold, is able to create high-accuracy structures for 87 out of 146 free modelling domains in the 14th Critical Assessment of Protein Structure Prediction (CASP14) benchmark [103]. The resulting dataset covers 58% of residues with a confident prediction, of which a subset (36% of all residues) have very high confidence. To give the reader a rough idea of the great impact of this research, it should be remarked that traditional techniques to elucidate the shape of a protein based on synchrotrons or on electron cryomicroscopy normally take months or even years. AlphaFold predicts the structure of a given protein in just a few minutes. Moreover, there are only about 180,000 protein structures available in public databases, but the authors intent to publish 100 million structures in the next few months. Actually, the final dataset already covers 98.5% of human proteins with a full-chain prediction. Thus, this contribution in protein

structure prediction is expected to accelerate research in virtually all areas in Biology.

In addition to DL, Reinforcement Learning (RL) is another renowned sub-discipline in ML. It pursues the goal of teaching an agent to interact with a given environment to maximize some notion of cumulative reward in the long term [151]. During training, a policy function that determines the action to be taken by the agent is optimized in an iterative trial and error learning process. RL has been living a renaissance in recent years [71] thanks to more powerful computers, new algorithmic techniques, mature software packages and architectures, and strong financial support [81]. Among those novel algorithmic techniques, it stands out the combination of DL and RL, giving rise to a new family of algorithms known as Deep Reinforcement Learning—henceforth Deep RL. These algorithms integrate ANNs in some of the basic components of a RL system such as the policy function, the value function, or the transition model. It should be highlighted, therefore, that Deep RL comprises algorithms that are essentially RL algorithms, so they share most of the pros and cons of this well known type of learning in ML. The only but substantial difference is that Deep RL algorithms make use of ANNs as function approximator in some of its component. Although there has been several important attempts to integrate DL and RL algorithms in a single system, nobody had been successful until the advent of striking breakthroughs such as Deep Q-network (DQN) [99, 100] and AlphaGo [140, 141]. Other remarkable advances have been made later to deal with traditional RL challenges like long time horizons, multi-agent settings, complex, continuous state-action spaces, and imperfect information [9, 161].

In fact, Deep RL has also lead to an extensive variety of applications such as autonomous driving, industry automation, natural language processing, healthcare, robotics manipulation, etc. RL and Deep RL have been applied in drug discovery as well in areas like molecular de novo design [102], retrosynthesis [133], and inverse-design chemistry [127]. Thus, Deep RL is expected to revolutionize the field of AI in the next

years [1] since it is considered as one of the closest approaches to the longed-for concept of Artificial General Intelligence (AGI) [124, 163].

Therefore, in the current doctoral thesis it is intended to take advantage of the promising algorithms of Deep RL to enhance one of the most popular in silico methods in drug discovery known as Docking. DL techniques based on supervised learning are already producing outstanding achievements in drug discovery and VS [66]. To the best of our knowledge, there had not been any attempt to apply Deep RL in Molecular Docking. Thus, we believe that the current research could contribute to accelerate drug discovery and be able to deliver medicines to patients in a shorter time frame.

1.2 Hypotheses and objectives

Solving the Docking problem is an expensive process from a computational point of view, as mentioned previously. In fact, the fastest traditional methods of Docking cannot process vast molecular databases in a feasible amount of time [54]. The central hypothesis of this dissertation, then, is formulated as follows:

- **Main hypothesis. The resolution of Docking can be enhanced by using an Artificial Intelligence approach based on Deep Reinforcement Learning algorithms**. In particular, the finding of the optimal pose for a given conformation can be accelerated in comparison with traditional Docking methods.

In addition to the central hypothesis, it is necessary to make the following assumption when defining the states of Deep RL to fully understand the research work behind this thesis and specially the structure of Chapter 3. In the context of this dissertation those states refer to the molecular representation/encoding. Those states are later used as input data for an ANN.

- **Secondary hypothesis. It may necessary to add structural information from both the ligand and the receptor to build the states of Deep RL**. This can correctly inform a model of ANN to solve the Protein-Ligand Docking Prediction problem.

Thus, the ultimate goal of the current dissertation can be deducted from the previous main hypothesis and stated in this wise:

- **Ultimate objective. To accelerate the resolution of the Protein-Ligand Docking Prediction problem for any given ligand-receptor pair compared with traditional Docking methods.**

Furthermore, the following specific objectives are proposed in order to accomplish this ambitious and challenging aim:

- **Specific objective 1. To set up the basic system based on a Deep Reinforcement Learning algorithm to solve the Protein-Ligand Docking Prediction problem**. Such system will be devised to incorporate a Deep RL algorithm yet to be determined—for instance, Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), or Asynchronous Advantage Actor Critic (A3C). Likewise, it should be flexible enough to accommodate different approaches with respect to the basic RL components. For instance, the user should be able to change the state representation, the size of the action space, the reward function, the model of the value/policy function, the rules of the environment, etc. This modularity is essential in a research context such as this one.

- **Specific objective 2. To implement the basic system created in the Specific objective 1 by representing the states through a feature vector to solve the Protein-Ligand Docking Prediction problem in a simple environment**. The initial Docking setting will be simplified as much as possible to quickly test the viability of the

system. Thus, a small receptor requiring less computation will be selected for testing. Moreover, a small feature vector will be used to represent the Docking scene. Such feature vector will contain positional information of the ligand like 3D coordinates and quaternions. As a consequence, a standard feedforward ANN will be used to approximate the true value/policy function.

- **Specific objective 3. To further develop the basic system created in the Specific objective 1 by representing the states through images to solve the Protein-Ligand Docking Prediction problem in a simple environment**. Following the secondary hypothesis, it will be necessary to add structural information from both the ligand and the receptor in the RL states to correctly inform the model (a neural network) to solve the problem. This directly affects to the way that molecules are represented in the Docking scene. For further information about molecular encoding, see Section 2.3. In particular, the RL states will be represented as images from 2D drawings of molecules generated by a molecular viewer software since they include the structure of both molecules and their relative distance from each other. In addition, a multi-view model will be applied to avoid molecular overlapping considering that images are two-dimensional by nature but the molecular space is three-dimensional.

1.3 Contributions and impact

The contributions of this dissertation can be summarized as follows:

- QN-Docking, a novel Molecular Docking method based on Deep RL, was successfully developed. It was built upon Q-learning using a single-layer feedforward neural network to train a ligand (the agent) to find its optimal interaction with respect to the receptor molecule. In addition, the corresponding environment of RL

and the reward function based on a force-field energy function were implemented. The proposed method was evaluated in an exemplary molecular scenario based on the kaempferol and beta-cyclodextrin. Results for the prediction phase showed that QN-Docking achieved 8× speedup compared to stochastic methods such as METADOCK 2, a novel high-throughput parallel metaheuristic software for Docking. **The work of QN-Docking was published in *Applied Soft Computing* [138]**.

- An early approach using images to represent the states of Deep RL was explored. Images of the Docking scene were generated in each timestep using PyMol [134]. Furthermore, the original model from QN-Docking based on a standard feedforward neural network was replaced by a Multi-View Convolutional Neural Network [148] originally designed to solve problems of 3D shape recognition. The new method was named MVDQN. Similar results to QN-Docking were obtained in the context of the kaempferol and beta-cyclodextrin. Nevertheless, the Specific objective 3 (to implement the core system created in the Specific objective 1 by representing the states through images) could not be fully attained due to: (1) Sampling inefficiency: generating images of the Docking scene in each timestep of the algorithm is too costly for PyMol, specially with large receptors. This drawback slows down the learning process in excess considering that images have to be constantly generated during training; (2) Molecular overlapping: this challenge may be solved by using a transparency effect manually applied to the receptor in the molecular visualizer when generating the different views of the Docking scene. However, this can hardly be applied automatically in PyMol for any given ligand-receptor pair because many of the generated images turn out to be defective or non-informative with respect to the Docking state. Even so, this approach is on par with respect to the obtained poses from the feature-vector-based perspective and served to get closer to achieve the Ultimate Objective defined in 1.2.

- A first approach using point clouds to represent the states of Deep RL is being investigated. In particular, 3D atomic coordinates plus the atom type were included in the input data to be fed to a DQN. The model of the new proposal, named DQN-Docking, was also adapted to these new input data. This was necessary because the standard feedforward neural networks cannot directly handle point clouds due to their irregular, unstructured, and unordered nature. More specifically, a combination of the PointNet++ model [118] and the DQN Dueling architecture [166] was developed to select the optimal action for the agent in each timestep. The energy function is currently being vectorized to deal with larger receptor for the sake of generalization.

- A novel molecular Docking method named METADOCK 2 was built. This method incorporates several novel features, such as (1) A ligand-dependent blind Docking approach that exhaustively scans the whole protein surface to detect novel allosteric sites (i.e. global optimization); (2) An optimization method to enable the use of a wide branch of metaheuristics; and (3) A heterogeneous implementation based on multicore CPUs and multiple graphics processing units. The PhD candidate collaborated with other members from the research team to publish the article in ***Bioinformatics*** **[54]**.

- An initial version of QN-Docking was presented by the PhD candidate in the International Conference on Parallel Processing (ICPP) held in Eugene, Oregon, in August 2018. This work was included in the **Proceedings of the 47th International Conference on Parallel Processing** [137].

- The latest advances in the research of this doctoral thesis with respect to the method based on images for molecular encoding (see Section 3.3) have been formatted as a scientific paper. The manuscript has been recently submitted to the editors of The

Journal of Supercomputing (Q2). We are currently awaiting for their acceptance.

- The PhD candidate was awarded in May 2019 with the first prize in the competition "Mi tesis en 3 minutos" (My thesis in 3 minutes) within the Doctoral Program in Computer Technologies at Universidad Católica de Murcia. The student defended his research work developed so far in front of the corresponding committee while showing a short video tutorial hosted on YouTube for that purpose.

1.4 Thesis structure

In this section, it is summarized the content of each chapter included in the current doctoral thesis:

- **Chapter 1: Introduction**. In this chapter it is performed the contextualization of this doctoral thesis by describing the addressed disciplines and fields, the hypotheses and objectives, the general contributions and research impact, and the general structure of the document.

- **Chapter 2: Background and related work**. This chapter includes the theoretical fundamentals of Molecular Docking and DQN necessary to fully understand the content of rest of the chapters in this dissertation. Moreover, it also contains a specific section to review the previous works of DL applied to Docking, which can be considered as the closest studies to the research line addressed in this thesis.

- **Chapter 3: Reinforcement Learning applications for solving the Protein-Ligand Docking Prediction problem**. This chapter is divided according to the specific objectives. More specifically, it is first described the conceptualization of the basic system mentioned in the Specific objective 1 to solve the PLDP problem. In particular, it is comprehensively explained the reformulation of the PLDP problem as a

RL task and the RL components—i.e. the agent, value function, environment, etc. Such system will be the cornerstone for next implementations of this chapter based on how the molecules are encode (or the RL states are represented): (1) Using a simplified feature vector; and (2) Images. The corresponding methods, experimental design, results, and discussion are presented for these two approaches.

- **Chapter 4: Discussion, conclusions, and future work**. Finally, this chapter includes a general discussion for the whole research work considering the results from the two different approaches to represent the states of the Docking scene with respect to the Deep RL algorithm. Likewise, the common conclusions and avenues for future research are also incorporated in this final chapter.

- **References**. In this section, the reader may find the bibliography in order to delve into the details of the different issues addressed in this dissertation.

Chapter 2

Background and related work

2.1 Molecular Docking

The PLDP problem involves two molecules known as the ligand and the host. The ligand—i.e. the pharmacological candidate or compound—is the smallest molecule with normally less than 200 atoms. The host, also known as the target or the receptor, is typically a protein or enzyme involved in a given disease [92, 125]. Molecular Docking is a computational method that models the interaction between the ligand and the host to solve the PLDP problem and has become an essential tool in drug discovery in recent years. The main goal in Docking is for the ligand to find the optimal interaction site where both molecules interact with one another (see Figure 2.1). To do so, Docking consists of two interrelated steps: (1) Sampling conformations of the ligand in the binding site of the host. The binding site, also called pocket site, encompasses the region on the protein that binds to the ligand with specificity; and (2) Accurate prediction of the interaction energy associated with those conformations using an energy function—henceforth Scoring Function (SF). In this definition, it is assumed that the location of the binding site of the protein is known. If the binding site is totally unknown, then the

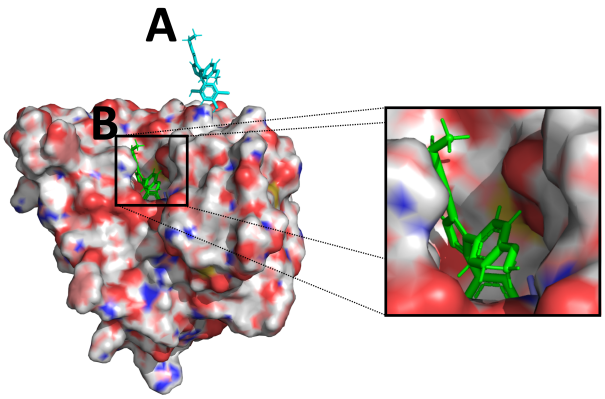

Figure 2.1: Illustration of the PLDP problem. The objective for the simulated ligand (A, with a blue skeleton) is to find the optimal interaction site (B, with a green skeleton) in the host surface. The optimal interaction site is zoomed for clarity sake.

problem is called Blind Docking [48, 110].

Moreover, there exist two different approaches in drug design, Structure-Based Drug Design (SBDD) and Ligand-Based Drug Design (LBDD) [33, 98]. The latter approach is also known as Similarity-Based Drug Discovery. SBDD exploits three-dimensional structural information gathered from the protein, while LBDD is based only on the knowledge implicitly contained in the chemical structure or physical properties of other ligands—i.e. their similarity—that bind to the biological target of interest. Needless to say, Molecular Docking falls in the SBDD category. In addition, Docking is performed multiple times for different chemical compounds from a ligand library in a method called Structure-Based Virtual Screening (SBVS). From this point on, the term of VS will be used in reference to SBVS particularly. Such a method broadly encompasses the following methods ranked from lower to higher accuracy: Docking, Molecular Dynamics (MD), and Quantum Mechanics (QM). Although MD and QM are far more precise than Docking when recreating the protein-ligand interactions, they are even more computationally expensive. This serious downside make these two methods impracticable in a VS context where thousands or even millions of candidates are normally tested.

In SBVS applied to Docking, large libraries of ligands are computationally screened

against a target of known structure, and those that are predicted to bind reasonably fine are experimentally tested through a subsequent High-Throughput Screening (HTC) process [75]. HTC is a pre-clinical method based on industrial robotic arms, data processing/control software, liquid handling devices, and sensitive detectors to quickly conduct millions of chemical, genetic, or pharmacological tests. Additionally, the tested libraries may contain millions of compounds [55]. So, the underlying assumption is that the more extensive and diverse the database, the higher possibilities of discovering new drugs. Nonetheless, SBVS methods currently fail to make accurate activity and toxicity predictions. This is due to constrains both in the capability of the SFs from a theoretical point of view and in the access to sufficient computational resources. The consequence of these drawbacks is that even the quickest SBVS methods are not able to process large chemical databases in a reasonable amount of time.

Another important distinction in Docking is related to flexibility of the involved molecules. Early Docking programs follow the lock-and-key theory [34], which conceives the ligand-host binding mechanism as a rigid ligand fitting into a rigid host just as much as a key fitting in a lock. A more realistic approach is that of based on the induced-fit theory [70], which states that both the ligand and the active site of the host are continually reshaped by the interactions between each other. However, adding flexibility to the host is a great challenge, especially with respect to backbone flexibility [47]. MD would be the ideal way of addressing this issue. Unfortunately, MD entails a much higher computational cost, as previously mentioned. This prevents MD from being routinely applied to screen vast biological databases. Consequently, in nearly every Docking software nowadays it is adopted an intermediate position that only considers the ligand as flexible. This last approach implies a good trade-off between accuracy and cost.

Thus, the resolution of the PLDP problem is not an easy, straightforward task. As for the conformational search, there are six degrees of translational and rotational freedom as

well as the conformational degrees of freedom of both the ligand and protein. Therefore, there is a huge number of potential interaction modes between two molecules, which makes Docking an NP-complete problem [139]. In other words, it is computationally unfeasible to generate all the possible conformations or to perform an exhaustive search. Still, if the best Docking pose—i.e. the one with the lowest SF value—were found, evaluating its binding energy would not be a trivial task either. At present, the accuracy of the SFs remains the main limitation affecting the reliability of Docking and SBVS [2, 75]. In fact, there are three types of SFs depending on the nature of the information included in their estimates: force-field-based, empirical, and knowledge-based [97]. More recently, it has emerged a new type of SF based on ML and DL (see Section 2.4 for further details). Each of those SFs have their own benefits and drawbacks.

Another serious problem with SBVS and Docking is that they cannot be applied to target molecules with unknown 3D structures. Unfortunately, proteins with known 3D structure are still relatively small compared with all the available chemical space [165]. Consequently, some authors like Rifaioglu et al. [123] argue that SBVS is not generally suitable for large-scale ligand-protein interaction prediction. It is also important to note that the available data for protein-ligand binding is inherently biased. For instance, the DUD-E benchmark dataset is reported to suffer from the negative selection bias problem [16]. In addition to molecular flexibility, there are other issues to bear in mind, such as the simulation of structural water located in deep cavities of the host, drug toxicity, or how to represent or encode the molecules involved in the PLDP problem. Last but not least, there is a natural limit of computational resources that makes necessary to find a good balance between accurate, realistic representation of the molecular interactions and the cost of such resources. In spite of all this difficulties and disadvantages, SBVS and Docking may narrow the search space down to few hundreds of compounds with desired properties to be further investigated in the subsequent stages of drug development. They

are not definitively the panacea but what is undeniable either is that nowadays they have become an essential part of drug discovery [66, 92, 114, 119].

2.2 Deep Reinforcement Learning and Deep Q-Networks

As previously mentioned, in RL an agent interacts with a certain environment trying to learn an optimal control policy to maximize the sum of some kind of cumulative reward. In finite-horizon, episodic tasks, the agent learns across a set of subsequent attempts or episodes made of several time-steps. As it is shown in Figure 2.2, the agent observes a state s_t at a given time-step t, takes an action a_t, gets the reward r_t from the environment, and transitions to the next state s_{t+1}. Unlike supervised learning, there is no explicit dataset. The environment generates the data according to the actions that the agent takes. Additionally, the proposed method is focused on non-stationary and Partially Observable Markov Decision Process—where the agent is only able to see part of the world state instead of the internal state (memory) to act optimally—problems.

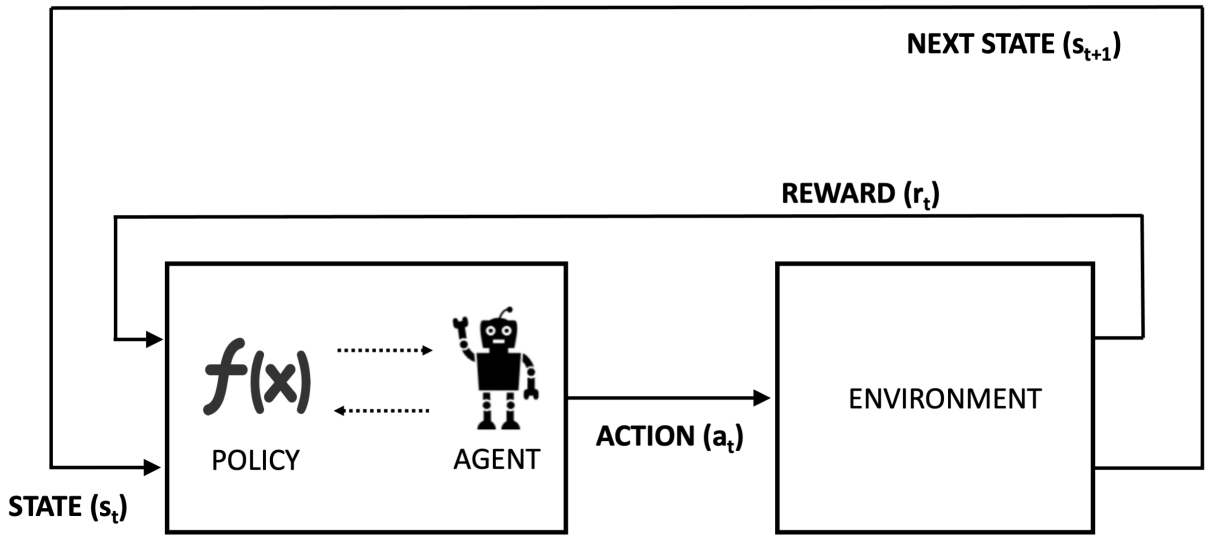

Figure 2.2: Representation of a typical problem in Reinforcement Learning.

RL is probably the hardest type of learning in ML in part because it encompasses many different problems, each of them with their own peculiarities. For example, Can the problem be modelled as a multi-armed bandit problem or as a Markov Decision Processs (MDP)? If it is the latter, is it a Full or a Partially Observable MDP? Environment dynamics are stationary or non-stationary? Is it a finite, indefinite, or infinite horizon task? If they are finite, Could be considered as episodic or non-episodic? Are the state and action spaces discrete or continuous? Does the size of the problem require a tabular or approximate solution? Is it more suitable to use model-based or model-free algorithms? If it is the latter case, Is it better to try optimizing the policy function directly or a value function? Which strategy is more appropriate to handle the exploration vs. exploitation trade-off? If a value-function-based algorithm is used, What degree of bootstrapping regarding Temporal Difference should be applied? Is the agent facing one or more tasks? Is there one agent interacting with the environment or are there several of them at the same time? Etc. This heterogeneous nature of RL gives rise to a plethora of algorithms, as shown in Figure 2.3. Describing every of the aforementioned theoretical concepts and each of those algorithms is beyond the purpose of this dissertation, so they are not covered in this section. For a further study of the fundamentals of RL, the reader is referred to Sutton and Barto [151].

Furthermore, DQN is the selected algorithm of Deep RL to build the system mentioned in the Specific objective 1 (see Section 1.2) that is expected to solve the PLDP problem. As shown in Figure 2.3, DQN is a MDP, model-free, value-based, off-policy algorithm. MDP is "a classical formalization of a sequential decision making problem, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards" [151]. By model-free, it means that DQN relies on real samples from the environment to take actions, as opposed to model-based methods that leverage models to provide the agent information about the dynamics of the

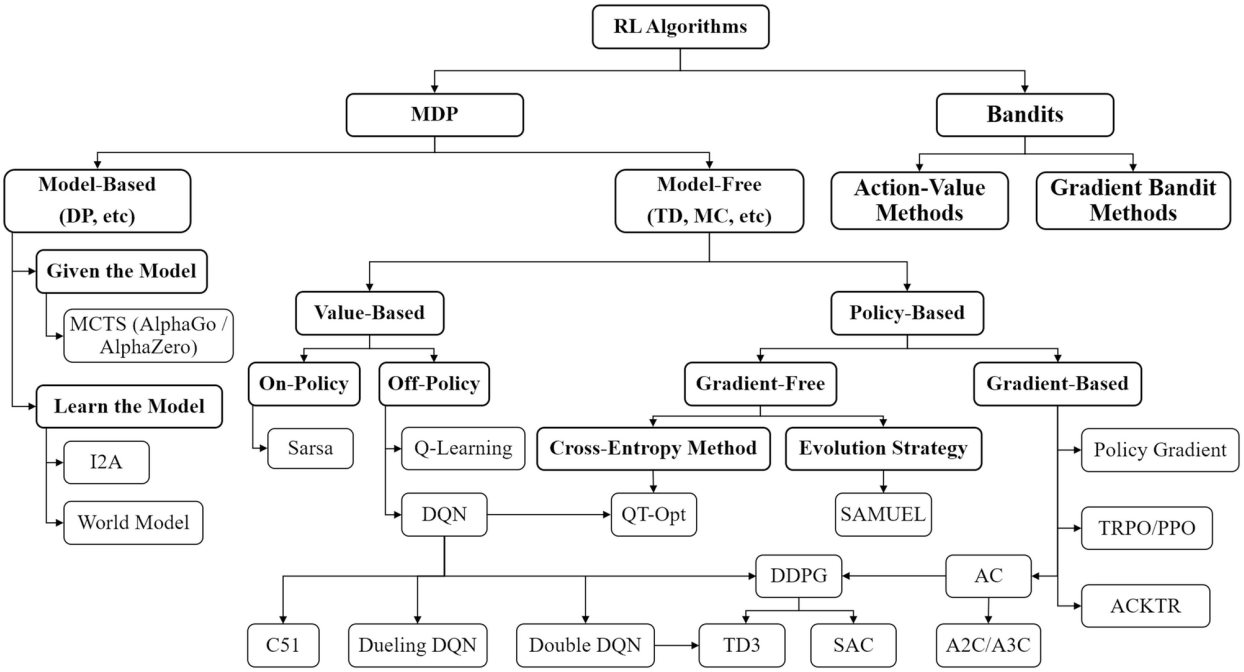

Figure 2.3: Map of Reinforcement Learning algorithms [174]. Boxes with thick lines denote different categories, others denote specific algorithms.

environment. In value-based methods, the policy function that tells the agent what action to choose next is not directly estimated as in policy-based methods. Instead, a value function is estimated (by selecting the action with the best value) and the policy function is derived from that value function. Finally, off-policy algorithms are those where the policy that is used to update the value function is different from the behavior policy used for acting. For a detailed justification of the choice of that algorithm, the reader may consult Section 3.1.

In the following lines this algorithm is briefly described, including its updating rule, the loss function for the ANN, additional artifices to favor convergence, and further improvements in recent years. DQN is based on another well-known algorithm called Q-learning [168]. Q-learning is a model-free algorithm because the transition probabilities are unknown. Instead, the environment produces the states and rewards. In addition, it is value-based since it tries to learn a state-action value function—Q-function from

here onward—that reflects the utility values (or Q values) of each state when executing a certain action, instead of directly learn the optimal policy as in policy-based algorithms. These utility values represent the estimated accumulated reward for the remaining time-steps in the episode. More specifically, those Q values are continuously updated according to the rule in Equation 2.1:

$$Q'(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{2.1}$$

where $Q(s,a)$ is the current expected utility of taking action a in the state s, α is the learning rate, and the term in parenthesis refers to the error between the target expected utility $r + \gamma \max_{a'} Q(s',a')$ and its current predicted value $Q(s,a)$. γ is a discount factor (normally close to 1) that is used under the assumption that estimated future rewards are worth less than certain immediate ones. The predicted expected utility is equal to the sum of the immediate reward r plus the discounted Q value of the next state s', assuming that the best possible actions is taken (a'). Q-learning is considered as an off-policy algorithm because the next action a' is selected to maximize the value of the next state s' instead of following the current policy—also known as the behavior policy—as in on-policy methods such as SARSA [150]. In particular, DQN follows an ϵ-greedy strategy as the behavior policy in order to manage the exploration/exploitation trade-off in RL.

DQN is also an approximate solution method, not tabular. In particular, an ANN is used to approximate the Q-value function $Q(s,a|\theta)$, where θ represents the weights from the ANN that parametrize those Q values. Those weights are updated iteratively during the training process. In every of those iterations, the goal is to minimize the loss function included in Equation 2.2:

$$L(s,a|\theta_i) = (r + \gamma \max_{a'} \hat{Q}(s',a'|\theta_i^-) - Q(s,a|\theta_i))^2 \tag{2.2}$$

The weights are updated in the next iteration $i+1$ based on backpropagation by computing $\theta_{i+1} = \theta_i - \alpha \nabla_\theta L(\theta_i)$. As the Q-function performs several weights updates and its estimates become more reliable, the agent starts to take more and more deterministic actions based on that function. Adam [67] is chosen as the update rule for QN-Docking. In supervised learning, the loss is the difference between the actual and the predicted values. Similarly, in the context of DQN the loss is the difference between the target values $r_t + \gamma \max_{a'} \hat{Q}(s', a'|\theta_i^-)$, and the predicted values $Q(s, a|\theta_i)$. Note that θ_i^- refers to the weights of the so-called target network, which is a copy of the deep Q-Network—also known as online network. The difference between both ANNs is that the weights of the target network normally remain fixed, being updated only every C iterations, while the online network is updated in every single iteration. In other words, in every iteration the estimated Q-values from the online network get a little closer to the ones from the target network. But the target network's weights are also updated now and then. Therefore, it is as the algorithm was chasing a moving target, hence its highly oscillated training process. It should be noted as well that the deep Q-Network architecture is based on a Convolutional Neural Network (CNN) comprised of three hidden layers—two convolutional layers (without pooling) along with a fully-connected layer [72, 82]. CNNs are a particular architecture of neural networks that performs specially well in image recognition.

In addition, the Q-learning algorithm with one-step return based on non-linear function approximators such as ANNs is known for its difficulty to converge. Thus, the authors of DQN adopted additional measures to enable and speed up convergence, such as the use of an experience replay database [87], the target network with frozen weights, and reward clipping. Namely, the experience replay dataset is used to store a fixed number of experiences or memories, which are transition tuples containing (s_t, a_t, r_t, s_{t+1})–i.e. the current state, the action taken, the reward obtained, and the next state. Those expe-

riences are uniformly sampled from the dataset in minibatches to train the ANN. The addition of those experiences helps to break the correlation between samples from subsequent time-steps, and therefore facilitates convergence.

Moreover, DQN has been improved over the years with several refinements [49]. Thus, double deep q-learning [159] tackles the problem of overoptimistic value estimates by evaluating the greedy policy according to the online network, while estimating its value following the target network. This modification improves DQN in terms of value accuracy and policy quality. In addition, the dueling network architecture [167] contributes to identify which states are valuable per se, without learning the effect of each action for each state. To do so, it includes two separate estimators, one for the state value function $V(s)$ and one for the action advantage function $A(s, a)$. The stream $V(s; \theta, \beta)$ of the model learns a general value that is shared across many similar actions at that state s, leading to a faster convergence. Finally, prioritized experience replay [130] also speeds convergence by sampling experiences according to how surprising or unexpected they are instead of doing it randomly. To avoid overfitting, though, a stochastic sampling method that interpolates between pure greedy prioritization and uniform random sampling is employed.

2.3 Molecular encoding

DL methods are already producing outstanding achievements in drug discovery and VS, as described in Section 2.4. This success may be explained, among other factors, by advancements in molecular and biological representation instead of using traditional human engineered descriptors. In effect, the interaction between protein and ligand is the root of the binding affinity. For this reason, it is necessary to painstakingly describe the protein–ligand interactions while reducing redundant information as much

as possible, which may largely determine model performance [21, 27]. Nevertheless, molecular encoding is a double-edged sword. As Kimber et al. [66] say, "the interactions between protein and ligands are complex, and encoding the most informative bits in a computer-readable format is one of the main challenges in both Cheminformatics and Bioinformatics." In the same vein, Bender and Cortes-Ciriano [7] point out that representing drug discovery information for AI is difficult because a single or even few descriptors of a compound are not able to anticipate the full biological complexity of drug effects, which, however, is the underlying assumption of many computational drug discovery approaches. In fact, there is no workhorse representation for each problem in drug discovery. With the coming of DL, different kinds of encoding formats to describe molecules can be inputted into variants of ANNs, which bring more choices for drug discovery. In the case of Deep RL, in addition, the size of the input data that fed to the ANN is especially sensitive since it can affect system performance dramatically. For all these reasons, it is briefly reviewed the main alternatives for molecular encoding in VS in the rest of this section. Actually, molecular encoding is the backbone of this dissertation, as the reader may deduct from the structure of Chapter 3. It should be highlighted as well that molecular representation is a burgeoning topic in drug discovery. More specifically, this section could not have been written so extensively for the simple reason that most of the cited works were published this year or last year (2020 and 2021).

Thus, Xu et al. [173] distinguish the following molecular representations in drug discovery: molecular descriptors, fingerprints, sequence data—such as Simplified Molecular Input Line Entry System (SMILES) and SMILES Arbitrary Target Specification (SMARTS)—, molecular graphs, and 2D and 3D structures of molecules. In the context of VS and protein-ligand interaction, Kimber et al. [66] identify different representation formats according to the molecules involved: ligand, protein, or complex (both the ligand and the protein) encoding. Following the recent overviews of Kimber et al. [66] and Qin

et al. [119], the rest of this section will focus on the molecular representation in the context of VS at a complex level, namely on molecular descriptors, Interaction FingerPrints, molecular graphs, 3D grids, and other encoding formats.

Molecular descriptors are domain-specific, physiologically relevant features traditionally used in ML models for drug discovery. In the context of SBVS, descriptors encode a binding interaction with numerical values describing the existence of certain binding features including their degree and frequency. Likewise, these chemical descriptors used to represent binding features can be zero-dimensional (if generated from a scalar), 1D, 2D, 3D or 4D [153]. They are extremely diverse, as they cover geometric descriptors (including coordinates, distances, angles, surface areas, and curvatures), chemical descriptors (such as atomic partial charges, Coulomb potentials, atomic electrostatic solvation energies, and polarizable multipolar electrostatics), and other advanced descriptors [14].

According to the way that protein–ligand interactions are represented, these descriptors can be grouped into knowledge-based descriptors, descriptors derived from terms of force-field or empirical SFs, and descriptors based on advanced mathematics [119]. Knowledge-based descriptors are usually estimated by counting the protein–ligand contacts between two elemental atom types or calculating the distance-dependent pairwise statistical potentials (see Figure 2.4). The force field-based SF utilizes classical force fields to calculate non-covalent protein–ligand interactions, such as Van der Waals and electrostatic interactions. The terms molecular mechanics PoissonBoltzmann surface area (MM-PBSA) and molecular mechanics generalized Born surface area (MM-GBSA) are normally included in the computation to deal with the solvation effect [88]. In addition, the empirical SF includes further terms representing entropic contributions, such as polar surface area and solvent-accessible surface area (SASA). Therefore, the terms in the force field and empirical SFs are potentially used to represent the binding features. As

for the descriptors based on advanced mathematics, there are three types of mathematical methods to represent binding interactions: differential geometry, algebraic topology, and graph theory. The fundamental hypothesis of these methods is that the physics and intrinsic properties lie in low-dimensional subspaces or manifolds embedded in a high-dimensional data space [106]. The models based on these descriptors have succeeded in the Drug Design Data Resource (D3R) Grand Challenges [78, 105]. For a comprehensive list of the computational models based on molecular descriptors to represent protein–ligand interactions, the reader is referred to Table 2 in Qin et al. [119].

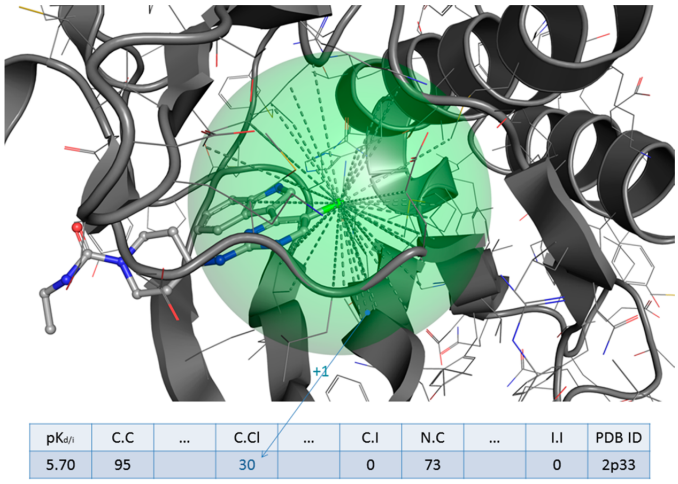

Figure 2.4: Sketch of the process of characterizing the protein-ligand complex (PDB: 2p33) as a set of structure-derived descriptors (C.C to I.I) in Ballester et al. [4]. The discontinuous green lines connect the ligand chlorine atom with all protein carbon atoms within the distance cutoff represented by the green sphere, with the number of these pairs giving value to the C.Cl descriptor. The rest of the descriptors are calculated in an analogous manner.

Interaction FingerPrints (IFPs) describe the interactions between a protein and a ligand based on a defined set of rules [152]. The bit string, in this context, encodes the presence or absence of interactions between the ligand and the surrounding protein residues. Some of these fingerprints are derived at the level of protein residues, while some others are derived at the atomic level [119]. In the former case, each binding site residue is described by the same number of features, which usually include interaction

types such as hydrophobic, hydrogen bond donor and acceptor. There exist several IFPs encoding interaction types such as Structural Interaction Fingerprints (SIFts) [24], Protein-Ligand Interaction Fingerprint (PyPLIF) [120], IChem's IFP [20], and PADIF [58]. For example, the SIFts represent the interactions between the ligand and binding site residues as an $n \times 7$ long bit string. In this case, the seven interaction types describe whether: (1) The residue, their main, or side chain atoms are in contact with the ligand; (2) A polar or apolar interaction is involved; and (3) The residue provides hydrogen bond acceptors or donors. Another example based on IChem's IFP is explained in Figure 2.5.

```
HBond LIG    OD1|31  |ASP 113-A  O17|1  |CAU  2.60 134.67
HBond PROT   ND2|99  |ASN 312-A  O17|1  |CAU  2.76 159.23
Hydrophobic  CZ3|60  |TRP 286-A  C16|2  |CAU  4.05
Hydrophobic  CZ |82  |PHE 289-A  C16|2  |CAU  3.74
Ionic LIG    OD2|32  |ASP 113-A  N19|4  |CAU  2.94
Ionic LIG    OD1|31  |ASP 113-A  N19|4  |CAU  3.60
Hydrophobic  CZ3|13  |TRP 109-A  C21|6  |CAU  4.43
Hydrophobic  CH2|14  |TRP 109-A  C22|7  |CAU  3.72
Hydrophobic  CZ |116 |TYR 316-A  C22|7  |CAU  4.46
Hydrophobic  CZ3|60  |TRP 286-A  C15|8  |CAU  4.06
Hydrophobic  CE2|81  |PHE 289-A  C15|8  |CAU  4.03
HBond LIG    OG |42  |SER 203-A  N7 |16 |CAU  3.31 127.77


|A W109|A D113|A S203|A W286|A F289|A N312|A Y316
10000000000101000010010000001000000000010001000000
```

Figure 2.5: Table of protein–ligand interactions and interaction fingerprint (IFP) generated by IChem [20]. For every ligand-binding residue, seven bits are switched either on (1) or off (0) as whether a particular interaction is detected or not with the ligand. Interactions are registered in a precise order (hydrophobic, aromatic face-to-face, aromatic edge-to-face, hydrogen bond accepted by ligand, hydrogen bond donated by ligand, ionic bond with ligand negatively charged, ionic bond with ligand positively charged).

The IFPs are sensitive to the order of the residues and may change according to the size of the molecules. This limits their application for ML purposes, as discussed in the last paragraph of this section. As a solution, Simple Ligand–Receptor Interaction Descriptor (SILIRID) [17] is proposed as a binding site independent and fixed-length IFP. In addition, distances with respect to interaction pairs or triples are introduced to explain the interactions more explicitly [25, 112, 129]. IFPs based on circular fingerprints—those that incorporate information about the arrangement of heavy atoms around each central

atom—are also applied with the aim of becoming more independent from predefined interaction types [19]. This is done by encoding all possible interaction types implicitly via the atom environment. For a thorough list of the computational models based on IFPs to represent protein–ligand interactions, the reader is referred to Table 1 in Qin et al. [119].

Moreover, it should be noted that fingerprints and molecular descriptors have been the traditional inputs of ML models for drug discovery, while other formats of molecular encoding such as sequence data and 2D and 3D structures were mainly used for storing and presenting molecules. Nonetheless, the design of fingerprints and descriptors is a complex and arduous process, so it cannot be performed at a large-scale level. Additionally, it can also entail the loss of essential information. Likewise, DL algorithms have made significant progress in representation learning, as explained in Section 1. They can directly learn highly nonlinear functions to minimize the input data. As a result, the latter formats have become more and more popular as encoding alternatives for ML and DL models. In the next paragraphs, these thriving formats of molecular representation are explained.

Complexes can be encoded using molecular graphs as well. These graphs represent the structural formula of a chemical complex comprising nodes (corresponding to the atoms of the complex) and edges (corresponding to the chemical bonds). The nodes normally come with an associated feature vector, while the bonds or the relationship between atoms are usually encoded in matrix form [66]. Some authors also consider features such as one-hot encoded atom type or degree and a binary value to describe aromaticity [86], for example. The node description in the complex is straightforward, but the complexity of molecular graphs in VS comes with the description of the interactions between the atoms. These interactions should include covalent and non-covalent bonds. Thus, Tsubaki et al. [157] use an attention mechanism to model those interactions

by computing which subsequences in the protein are more important for the ligand by assigning greater weights to the subsequences. This add some flexibility to capture the interactions between compounds and proteins rather than obtaining a simple summation.

Lim et al. [86] create another DL approach for predicting drug-target interaction using a distance-aware graph attention algorithm to differentiate various types of intermolecular interactions. In this case, they considers two adjacency matrices. The first one represents purely covalent interactions, while the second one describes both covalent interactions and noncovalent intermolecular interactions plus their strength through distances. Finally, Feinberg et al. [32] introduce another method for protein-ligand binding affinity prediction based on a Graph Convolutional Neural Network (GCNN)–see Figure 2.6. They set two tensors to treat the interactions between atoms: (1) a 2D tensor with the distance for each atom pair from the ligand and receptor; and (2) a 3D tensor whose shape is $N \times N \times N_{et}$, where N is the number of atoms and N_{et} is the number of edge types. So $A_{ijk} = 1$ if atom j is in the neighborhood of atom i and if k is the bond type between them. If not, that same entry is equal to 0. This procedure numerically encodes the spatial graph as well as the bonds through edge type.

Another type of encoding used in predicting protein-ligand interactions are 3D grids. In this format, the protein is embedded into a three-dimensional Cartesian grid centered on the binding site. Analogously to images, each little cube of the grid holds one or several values that describe the physicochemical properties of the complex at that specific position in the three-dimensional space. Such cubes may comprise just 1D floating point array [162] or a 4D tensor [147] to be used as input for a DL model, for example. The entire grid has normally a size between 16 and 32 ångströms (Å), while each cube is usually 0.5 - 1 Å wide [35, 149, 162]. The features included in each cube may consist of (1) Simple annotations of atom types and IFPs, as in Hochuli et al. [50], Li et al.

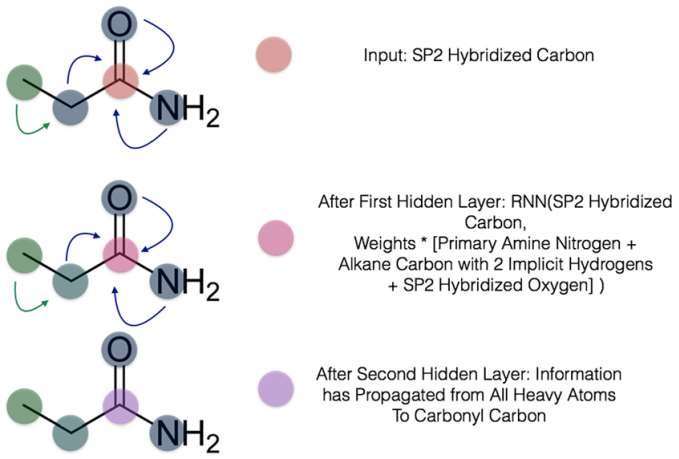

Figure 2.6: Visual depiction of the gated graph neural network with atoms as nodes and bonds as edges in Feinberg et al. [32]. The small molecule propanamide is chosen to illustrate the propagation of information among the different update layers of the network.

[80], Wallach et al. [162]; (2) Pharmacophoric or physicochemical features, as in Skalic et al. [143, 144], Stepniewska-Dziubinska et al. [147]; and (3) Energy-based attributes using one or diverse probe atoms, as in Erdas-Cicek et al. [31], Sunseri et al. [149]. An example based on pharmacophoric/physicochemical features of the ligand is shown in Figure 2.7.

Nevertheless, this approach based on 3D grids has also its own disadvantages. Although the size of the grid and its resolution may be adjusted, in general, it entails many data points to be used as input data for an ANN, which can slow down the training process in DL. Also, in the grid representation, a large amount of empty grid points can cause unnecessary computation and memory usage. Moreover, the grid representation can lose distance information between atoms depending on the selected grid spacing.

There are also other encoding formats in VS, which will only be shortly explained in the following lines. The first format is known as topology-based methods [13]. The study of topology deals with the connectivity of different components in a given space, and characterizes independent entities, rings and higher dimensional faces within the

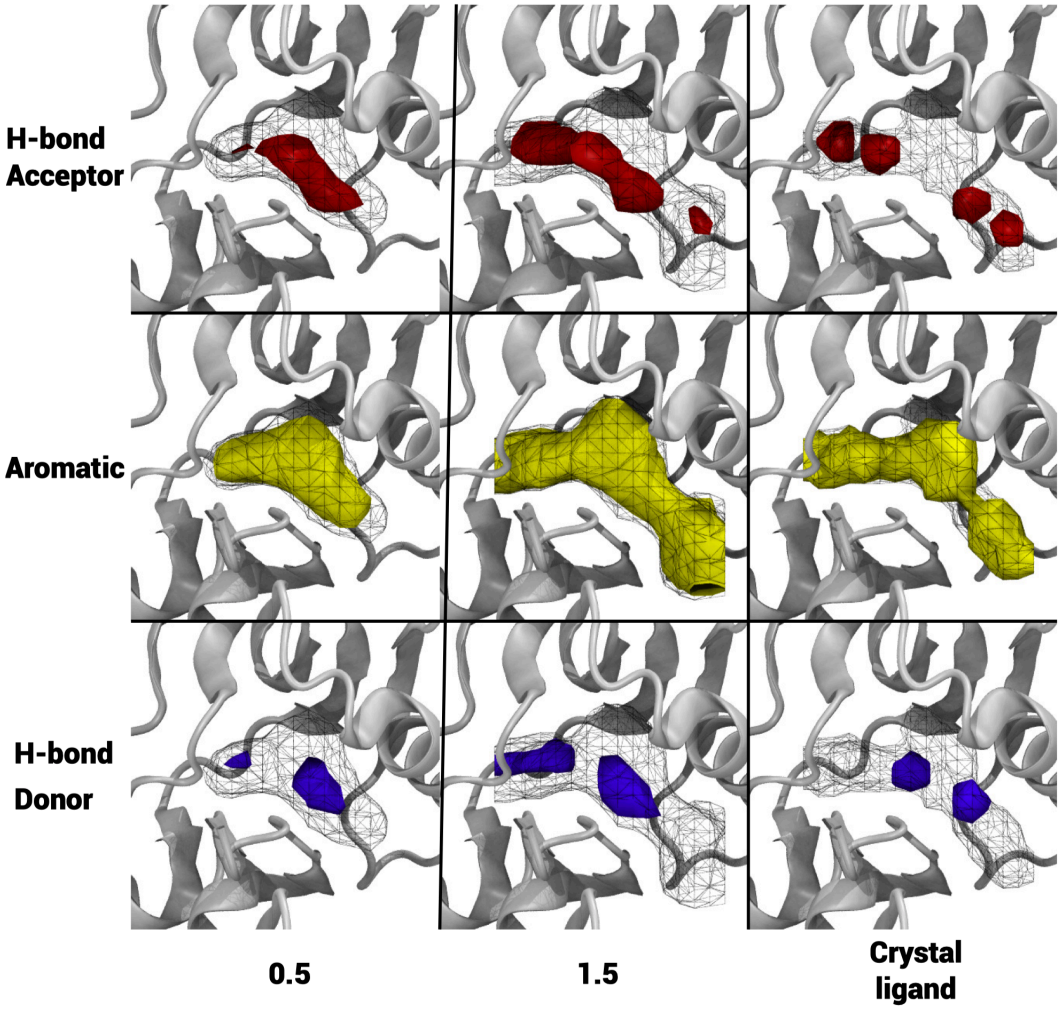

Figure 2.7: Example of generated properties (hydrogen bond acceptor, donor, and aromaticity) for (PDB entry 1FPU) in Skalic et al. [143]. Ligand occupancy is displayed in black wireframe while aromatics, Hydrogen bond acceptors and donors are in yellow, red and violet, respectively. The generated predictions shown in the column labeled 0.5 used half the atom counts of the cocrystallized ligand as the input. Column 1.5 follows the same logic, while the third shows actual cocrystallized ligand. As the atom count grows, the generated fields expand and are able to match more peripheral groups.

space. Although some authors like Kimber et al. [66] consider this type of molecular representation an independent format, for some others scholars they are a specific kind of fingerprints called "topological fingerprints" [13, 119]. Either way, element-specific topological invariants can help retaining the 3D biological information. In this case, the complex can be represented by an image-like topological representation, as if they were barcodes (see Figure 2.8). For example, Zhu et al. [176] introduce a molecular encoding simply based on the protein-ligand atom pairs and their distances. All atom pair energy contributions are summed, while the contributions themselves are learned through an ANN considering the properties of the two atoms and their distances.

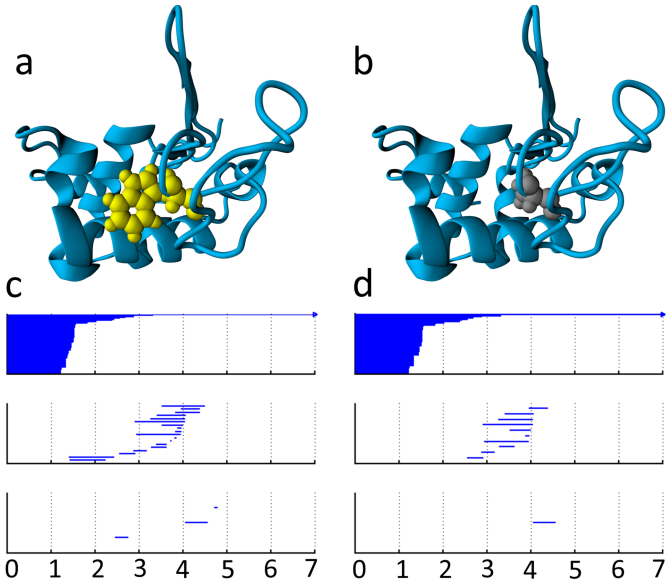

Figure 2.8: An illustration of barcode changes from wild type to mutant proteins from Cang and Wei [13]. (**a**) The wild type protein (PDB:1hmk) with residue 60 as Trp. (**b**) The mutant with residue 60 as Ala. (**c**) Wild type protein barcodes for heavy atoms within 6 Å of the mutation site. Three panels from top to bottom are Betti-0, Betti-1, and Betti-2 barcodes, respectively. The horizontal axis is the filtration radius (Å). (**d**) Mutant protein barcodes obtained similarly to those of the wild type.

Likewise, Pereira et al. [111] propose a Docking-based VS method based on DL. In this case, the interactions between atoms are handled through the concept of atom context—i.e. atom and amino acid embeddings. The idea of using information from

closest neighbor atoms of both compound and protein have been successfully explored in previous work on structure-based drug design [169]. The model is composed of an embedding layer that extracts relevant features from the atom context followed by a convolutional layer to summarized all that information in a fixed-length vector. The basic features extracted from the context of an atom include the atom types, atomic partial charges, amino acid types, and the distances from neighbors to the reference atom. Then, the first hidden layer in their ANN transforms each basic feature value of atom contexts into real-valued vectors (embeddings) by a lookup table operation.

There are also other encoding formats that have been successfully applied in drug discovery but barely used in the specific context of VS. That is the case of Simplified Molecular Input Line Entry System (SMILES) and 2D structures of molecules (2D grids) Xu et al. [173]. Such methods could be a promising alternative to accurately represent the protein–ligand interactions in an efficient manner.

Thus, sequence data have the advantage of being able to be used as input for more powerful variants of ANNs other than feedforward neural networks, such as Recurrent neural networks (RNNs) and CNNs. Simplified Molecular Input Line Entry System (SMILES) is a specification in the form of a line notation for unambiguously describing the structure of chemical molecules using short ASCII strings. It was developed in 1988 by [170] and has been the most popular line notation and sequence data ever since. The SMILES representation, non-unique and unambiguous, is obtained by assigning a number to each atom in the molecule and then traversing the molecular graph using that order and a given algorithm such as depth-first search, for example [22]. There can be multiple atom numberings for a given molecule, leading to different SMILES. The ensemble of SMILES representing one molecule can be referred to as enumerated or randomized SMILES and are obtained by, for each molecule, randomly selecting an initial node for graph traversal while keeping the same graph traversal algorithm, thus

leading to different atom orderings. To avoid conflicting SMILES representations for the same molecule, a unique SMILES can be designated, and several canonicalization methods exist to this end [108, 132]. SMILES has been successfully used to generate new SMILES sequences so that automatic de novo drug design can be realized [10, 39, 44]. It has been also employed with RL and Deep RL algorithms in de novo drug design [42, 107, 115, 127], as mentioned in Section 2.4.

Images have been also used to model molecules in drug discovery. Thus, Goh et al. [36] propose using images of 2D drawings of molecules to predict toxicity, activity, and solvation properties. The method is later improved by increasing the channel of feature vectors [37]. Their results show that intuitive information about molecules can be appropriate inputs of CNNs. In addition, Matsuzaka and Uesawa [94] develop a molecular image-based Quantitative Structure-Activity Relationship (QSAR) approach also based on images as inputs for their ANN. Unlike the previous work, they use 3D chemical structures generated by Molecular Operating Environment (MOE) software from SMILES, not from the 2D drawings of molecules directly. In a similar vein, Rifaioglu et al. [123] also use 200-by-200 pixel 2D images generated from SMILES sequences in the context of Docking.

Finally, it is worth to make three general considerations with respect to molecular encoding applied to VS. First, ANNs and their variants cannot accept variable-sized molecules as direct inputs. But ligands and proteins do vary in size. As a result, the encoding formats described above deal with this problem in several ways. For example, the fingerprint format of SILIRID is obtained from binary IFPs by summing up the bits corresponding to identical amino acids. This results in a fixed size vector of 168 integer numbers corresponding to the product of the number of entries (20 amino acids and one cofactor) and 8 interaction types per amino acid (hydrophobic, aromatic face to face, aromatic edge to face, H-bond donated by the protein, H-bond donated by the

ligand, ionic bond with protein cation and protein anion, and interaction with metal ion). Likewise, most of graph-based methods commented above have shared weights and biases for local atoms—similar to the convolutions in CNNs—to handle the fixed size problem. For instance, Tsubaki et al. [157] update vectors of nodes and edges by considering the surrounding nodes and edges within a certain radius. Lastly, 3D grids has a fixed size between 16 and 32 Å and normally limited to the binding site.

Second, many of the models behind the cited works in this section seek to replace traditional energy functions or SF for predicting binding affinity. These functions normally consider all the atoms from both the ligand and receptor to compute the binding score. The 3d-grids-based studies mentioned above or the work from Pereira et al. [111] relying on the atom context, however, only take into account the structure of the binding site. Therefore, it is still not clear whether these approaches are sufficient to accurately predict binding affinity compared to more traditional energy functions. And third, as the reader may deduce from the overviews mentioned above, for the moment there is no clear winner with respect to molecular encoding in the context of Docking and DL. Actually, it is a topic continuously evolving and many papers with different encoding alternatives are released every year.

2.4 Related work

The summary of this section is displayed in Figure 2.9. As far as we are aware, there are barely peer-reviewed works in the intersection of Docking and Deep RL. There are published articles in other drug development problems like de novo drug design [57, 89, 102, 107, 115, 146] but this challenge and its difficulties are quite different compared to the Docking perspective.

Perhaps the closest study to the proposed approach would be that of the model

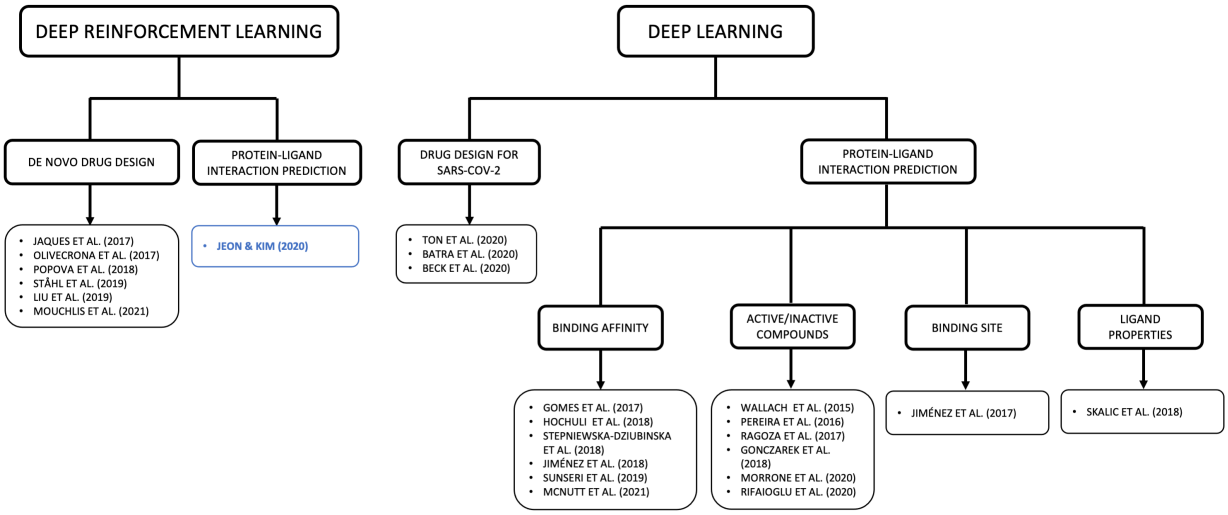

Figure 2.9: Overview of the performed literature review. The closest works to the intersection of Docking and Deep Reinforcement Learning are those in Deep Learning applied to Protein-Ligand Interaction Prediction.

named Molecule Optimization by Reinforcement Learning and Docking (MORLD) proposed by Jeon and Kim [59], as shown in Figure 2.9 in blue. MORLD entails a hybrid approach halfway between LBDD and SBDD. This integration of both perspectives have been proposed in the past by several research groups [3, 160]. In the case of MORLD, it takes an initial ligand in the binding spot of a given protein and generates a new pharmacological candidate in each episode of RL by adding or removing atoms and bonds as long as the valence constraints are satisfied. Then, the modified ligand is evaluated by LBDD metrics like Quantitative Estimate of Drug-likeness (QED) and the synthetic accessibility (SA) in every step. QED score is a quantitative estimate for how similar particular molecules are to the known drugs in terms of several physicochemical properties and structural features. SA score is used to estimate the ease of synthesis of drug-like molecules. The weighted sum of these two metrics is given as the reward for a DQN. If the episode reaches the final state where these two metrics cannot be further improved, then a Docking SF based on QuickVina 2 with respect to the target protein is computed and used in the RL system as reward.

Eventually, after sufficient number of episodes, MORLD tends to generate potential novel inhibitors with lower binding energy along with high SA and QED scores to the given protein structure. However, it remains unclear how switching the reward signal based on QED and SA at the end of the episode by the Docking score affects training since in MDPs is normally assume a fixed reward function. In addition, in this model it is assumed that the binding location is known. Therefore, this process could be considered more like some sort of fine-grained Docking. In contrast, the proposed methods in this dissertation solely rely on a SBDD approach to identify the optimal location inside the binding site.

Next, it is succinctly reviewed those works of DL applied to Docking since they could be considered as the closest ones to the intersection of the PLDP problem and Deep RL. Note, however, that these articles are based on supervised learning instead of RL. Over the last decade, a wave of DL methods and applications to boost VS has emerged [66, 113]. Such trend has become specially prolific in the last year due to the high interest in finding a remedy, either a novel vaccine or a small pharmacological molecule, for the SARS-CoV-2 and the global pandemic that is causing thousands of deaths around the world [5, 6, 154]. This development relies not only on new DL algorithms but on the availability of more and more compounds, structures and mapped bioactivity data, and novel encoding techniques. Basically, these works aim to substitute traditional Docking SFs and ML SFs by DL SFs to predict binding affinity [38, 50, 62, 96, 147, 149], active/inactive molecules [40, 101, 111, 121, 123, 162], binding sites [61], or properties of potential ligands interacting with proteins [143]. In the next paragraphs, the models, results, and some disadvantages of these studies are briefly reviewed.

In the models behind these works, 3D grids are the most widely used encoding format to describe the molecules. These 3D grids does not only take into account the atom coordinates but also extra information such as the atom types, partial charges, phar-

macophoric and SMART properties, voxel occupancy, atom connections, hybridization, amino acid types, etc. For these grids, 3D CNNs [60] is the preferred ANN's architecture. It is worth noting that only Jiménez et al. [62] make use of deeper ANNs—in particular a variant of SqueezeNet [51]—, while the rest of the authors make use of shallower CNNs with no more than three convolutional-pooling layers and a fully connected network with no hidden layers attached at the end, in most cases.

Results obtained up to now are slightly better or at par than other methods based on traditional and other ML SFs. For example, Wallach et al. [162] achieve an AUC greater than 0.9 in the DUD-E benchmark, surpassing previous Docking methods included in Smina [69]. But these results are constrained to 57.8% of the targets included in the benchmark. Furthermore, the CNN from Ragoza et al. [121] outperforms Autodock Vina [156] empirical SF, and ML-based SFs like RF-Score and NNScore in VS and inter-target evaluations of pose prediction. However, it performs worse at intratarget pose ranking, which is more relevant to Docking. Sunseri et al. [149] evaluate their CNN in the D3R challenge. They find that their performance is best-in-class when performing affinity ranking for two of the targets (three of the subchallenges), albeit it is average on two of the other targets and poor on a third. The model from Stepniewska-Dziubinska et al. [147] outperforms the SFs tested in Li et al. [79] in two test sets (PDBbind v. 2016 and CASF-2013)—the best-performing X-Score had R = 0.61 and SD = 1.78, while their model achieved R = 0.70 and SD = 1.61. However, the RF-Score v3 SF has better performance, achieving R = 0.74 and SD = 1.51 on CASF-2013. Jiménez et al. [62] compare their model with other three ML SFs (RF-Score, X-Score, and cyScore). In the PDBbind core set, their model is able to outperform the rest of the methods, with a similar correlation coefficient as RF-Score while achieving significantly lower error in terms of RMSE. In the CSAR sets, however, RF-Score offers the best average performance, supporting the hypothesis that more complex ML methods tend to underperform outside the training manifold.

McNutt et al. [96] use an ensemble of CNNs to outperform AutoDock Vina on reDocking and cross-Docking tasks both when the binding site is defined (Top1—the percentage of targets where the top pose is better than 2Å root mean square deviation—increases from 58% to 73% and from 27% to 37%, respectively) and when performing a Blind Docking (Top1 increases from 31% to 38% and from 12% to 16%, respectively). However, they do not compare their system with other Docking methods other than Vina. With respect to the binary classification problem of active/inactive compounds, Morrone et al. [101] proposed a graph-based CNN that includes a dual architecture based on separate subnetworks for the ligand bonded topology and the ligand-protein contact map. Their model for binding mode prediction uses Docking ranking as input in combination with Docking structures. It also outperforms AutoDock Vina in a variety of tests, including on cross-Docking data sets that mimic real-world Docking use cases. For example, on an independent test set drawn from the PDBbind database, their model gets an Area Under the Curve (AUC) = 0.90 and a success rate (the fraction of ranked "1" binding modes that are correct) = 0.380 against Vina's AUC = 0.66 and success rate = 0.364). Again, these authors only compared their method with Vina. Finally, Rifaioglu et al. [123] suggest another system based on deep CNNs for drug-target interaction prediction. Their system employs 2D structural representations of compounds based on SMILES. They tested their model on the DUD-E dataset and obtained a mean performance of AUC = 0.85, similar to Gonczarek et al. [40], Ragoza et al. [121], Wallach et al. [162].

Indeed, the main drawback of all these models is that they resoundingly fail when predicting the output with new examples out of the datasets they were trained for. This problem of generalization is not only related with the algorithms themselves and the molecular/input representation but also with the lack of gold-standard datasets in ML applied to drug discovery [122]. One of the difficulties is that the bioactivity and structural information need to be linked for solving the PLDP problem. Therefore, the

3D structural information of protein-ligand complexes stored in the Protein Data Bank (PDB) or the labeled bioactivity data available in PubChem and ChEMBL databases is not sufficient for Docking computational purposes. Another difficulty lies in the heterogeneity of the information included in the drug discovery datasets. In this sense, there have been some recent, laudable efforts in drug discovery to create something similar to ImageNet database in image recognition [35, 171]. For more information about the available databases in SBVS, the reader is referred to Kimber et al. [66]. In the meantime, datasets like PDBbind, scPDB, CSAR, DUD, and DUD-E remain as probably the most widely used benchmarks in protein-ligand interactions prediction. Finally, although some of these results obtained with CNNs are encouraging, some authors like Chen et al. [15] maintain that the problem of generalization casts doubts on the ability of these DL models to consistently improve results compared to traditional SFs and other ML-based methods.

Chapter 3

Reinforcement Learning applications to solve the Docking problem

As stated in Section 1.2, the ultimate goal of this dissertation is to accelerate the resolution of the Protein-Ligand Docking Prediction problem for any given ligand-receptor pair compared with traditional Docking methods. To do so, it is first necessary to build a versatile system capable of accommodating different interpretations regarding the formulation of Docking as an RL problem, as stated in the Specific objective 1. Second, following the secondary hypothesis of this dissertation, it is necessary to add structural information from both the ligand and the receptor to correctly inform the ANN to solve the PLDP problem. Therefore, two different alternatives for molecule representation are proposed next in this section: feature vectors and images—as stated in the Specific objective 2 and 3, respectively. In the next lines, the corresponding methods, experimental design, and results for each of these approaches are presented.

3.1 Basic system of Deep RL applied to Docking

In the following section it is described the core system mentioned in the Specific objective 1 in Section 1.2. This basic system will be shared by all the alternative models proposed in the Subsection 3.2 and 3.3, so it is worth to thoroughly explain its foundations.

As explained in Section 2.2, DQN is the selected algorithm of Deep RL to build the system. Before delving into the implementation of this algorithm, a justification for its selection is elaborated in the next lines. Among the MDP family, a model-free algorithm was chosen to redefine the PLDP problem as one of RL. This decision was made because of the absence of an accurate model to explain the dynamics of the environment—i.e. the full distribution of next states and next rewards. It should be noted that in the RL literature it is normally differentiated between the term "model" as a model of the environment that precisely provides dynamics of the environment, and the use of statistical learners such as ANNs, which are called "function approximators". In this dissertation, the term "model" will be used to refer to the ANN since there is no model for the environment dynamics. Within the model-free approach, a Q-learning perspective was selected since this kind of algorithms are usually off-policy, so they are prone to be more sample efficient than policy-based optimization algorithms because they can exploit data from experts or other sources e.g. in the case of DQN, that source would be the experience replay buffer. That pool allows to select the tuples of experiences to train the ANN without having to make use of expensive model calculations.

Next, it is first illustrated how the PLDP problem has been reformulated as an RL task. This brings about important decisions on the building blocks of RL and ANN–i.e. defining the states, actions, reward function, architecture of the ANN, stop conditions, etc. The major components of RL (see Section 2.2) associated with the PLDP problem are listed in Table 3.1. Those components are further explained below. Likewise, the

functioning of the general approach proposed in this dissertation can be visualized at a glance in Figure 3.1. As mentioned before, the system is highly inspired by the DQN algorithm, originally used to teach an AI's agent to proficiently play classic video games from the Atari 2600 console. In the context of Docking, the agent would be the ligand instead of the player. In Artificial Intelligence, an intelligent agent is anything which perceives its environment, is able to take actions autonomously to reach goals, and can improve its performance through learning or by using some kind of knowledge. Agents may be simple or complex. For example, a robot, a video game character, devices such as a thermostat, or even a human being or any system that meets the definition, like a firm, state, or biome may be considered as an agent [126]. The possible actions for this agent would be to move and rotate in the three-dimensional molecular space, although other actions such as molecular folding could be considered as well. Each time the agent performs an action from a given state, it receives a reward from the environment that could be related to a traditional energy function. The agent transitions to the next state or position and a new loop begins. The crucial question here is how to guide the agent to take the best actions to reach the global optimum. That role is performed by the policy function or indirectly by the value function. Thus, the value function in this case is estimated by a function approximator based on a Deep Q-Network.

As for the actions, at each time-step the agent takes a specific action a_t from the space of possible actions, $A = \{1, \ldots, K\}$. These actions include both translation and rotation. An important assumption in this doctoral thesis is that the physical or molecular space is considered as discrete. Consequently, in each timestep the ligand moves a specific distance (0.1 Å) and certain degrees of rotation (0.5 degrees). Therefore, the ideal size for this discrete action space is twice the degrees of freedom of moving the agent in the three-dimensional space. To do so, 6 degrees of freedom are involved, which give a total of 12 possible actions: translation and rotation in the three axes

RL component	Representation
Agent	Ligand
Value/Policy function	ANN estimating the Q values
Actions	Moving and rotating in one or three axes forwards and backwards
Environment	DockingEnv-v0
States	Feature vector or images
Reward function	Transformed score from a given SF

Table 3.1: RL components in the Docking problem.

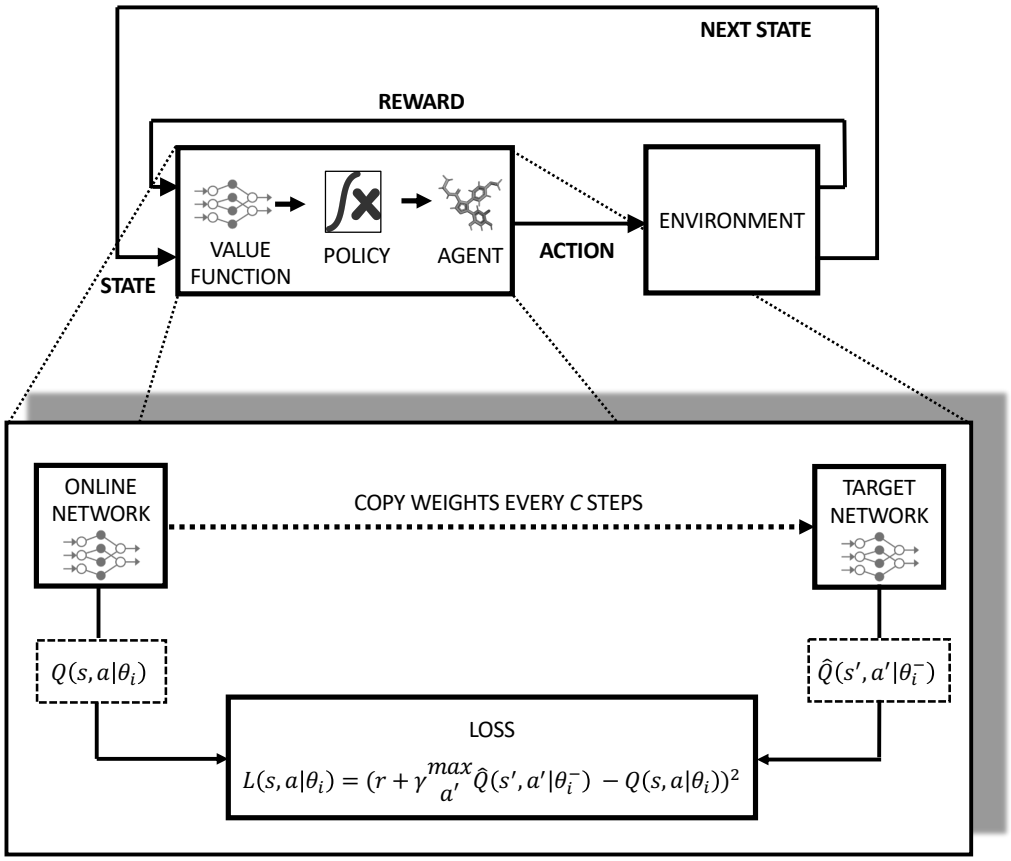

Figure 3.1: Operational schema of the general approach proposed in this dissertation.

forwards and backwards. Then, the selected action is used to output the new position or state of the ligand in the environment. For the RL environment ε, an OpenAI Gym

environment called DockingEnv-v0 is created, including the step, reset, and render methods for the corresponding class in Python. OpenAI Gym [11] is a Python's library that provides an easy to set up, general-intelligence benchmark with a wide variety of different environments—similar to the database offered in the ImageNet Large Scale Visual Recognition Challenge in supervised learning.

With respect to the states, these are directly linked to the molecular representation of the Docking scene. The alternatives for molecular encoding are diverse, as shown in Section 2.3, and there is no clear winner. Thus, two different approaches are proposed in Section 3: feature vectors and images.

Furthermore, the reward function is one of the most sensitive parts in RL since it serves as a guide for the agent to interact with the environment. It implies a deep knowledge concerning the problem to be solved–in this case, the PLDP problem. A first, natural attempt in this context is to use a traditional energy function or SF in Docking. Nevertheless, this approach is not valid, as explained in Section 3.2.1. The score from the SF turns out to be too noisy to be employed as a reward signal and it does not favor convergence of the RL algorithm. Alternatively, numerous prototypes of reward functions are tested along Section 3.

The ANN architecture is conditional upon the form of the states. Thus, a different architecture is used in each approach described in Section 3. Namely, in the case of feature vectors, a standard feedforward ANN is used as the function approximator for the value function. For images, a Multi-View Convolutional Neural Network (MVCNN) [148] is used in conjunction with the dueling network architecture [167]. These ANNs take transition tuples containing (s_t, a_t, r_t, s_{t+1})–i.e. the current state, the taken action, the obtained reward, and the next state–as input from the experience replay dataset. Moreover, those memories are sampled from the dataset in minibatches to train the ANN. Specifically, the criterion of absolute Temporal-Difference (TD) error is followed to relatively favor more

surprising (better) experiences. In turn, the network outputs the estimated Q values of each action in a given time-step. The action with the highest Q value is selected for the agent at that particular time-step.

Finally, another important aspect in RL refers to stop conditions. The Docking problem in this doctoral thesis has been conceived as a finite-horizon, episodic task. Consequently, it must be specified when a given episode is over. Thus, two stop conditions are manually added. First, a maximum number of time-steps (1,000) is set as in any RL episodic task. Second, sometimes the ligand may deviate and get away from the host excessively. To correct such undesired behavior, the episode immediately terminates if the Euclidean distance between the two molecules is greater than a certain cut-off D in the 10 following time-steps. In practice, this condition limits the exploration area of the agent around the host to the binding site. These two stop conditions definitely contribute to accelerate the learning process. Furthermore, an overall condition is necessary to determine when the agent has optimized the policy well enough. So, another condition is set through a callback function to stop the whole training process when the average reward in the last 100 episodes reaches a theoretical maximum. That quantity is calculated considering the distance between the initial position of the ligand and the optimal solution, the maximum time-steps per episode, the highest possible reward per time-step, etc. In the proposed methods, the aforementioned distance between the initial position and the solution can be estimated since the latter is known. In practice, however, the solution is unknown, so the theoretical maximum should be reformulated.

As a conclusion, in the lines above it has been devised a basic system with an embedded ANN to train the ligand to look for the optimal solution in a molecular Docking setting by following a reward signal derived from a traditional force-field-based SF. The proposed method is general and modular enough to be able to adapt to different interpretations of the the basic RL components described in 3.1. Such core system will

allow to incorporate different alternatives of state representation such as feature vectors and images (as intended in the next subsections) to solve the PLDP problem. Therefore, the Specific objective 1 has been successfully met.

3.2 Molecular encoding with a simplified feature vector

In this first approach, it is intended to achieve the Specific objective 2 stated in Section 1.2. In particular, it is expected to implement the basic system developed in the previous subsection to solve the PLDP problem. In particular, the proposed implementation is based on a feature vector to represent the states—namely, positional information of the ligand like 3D coordinates and quaternions. Thus, this first functional version, called QN-Docking, is heavily inspired by the DQN algorithm. Nevertheless, it does not make use of a deep ANN (3 or more layers) since the feature vector is too small to require such a powerful model. Specifically, it is based on a feedforward ANN with 1 layer with 256 units, as explained below. In this sense, the simplicity of the model to quickly assure the viability of the system is prioritized over the use of more advanced and complex models such as deep ANNs. Therefore, this first implementation cannot be considered as a Deep RL solution from a strictly technical point of view. Even so, this first method is a success and the Specific objective 2 is accomplished.

3.2.1 Specific methodology

To fully understand the current section, the reader is referred to subsection 3.1 about the foundations of the core system based on Deep RL to solve the Docking problem. Thus, Table 3.1 is updated with the assumptions made for the initial system based on a simplified feature vector as input data. The new table is listed as Table 3.2. It is worth noting that in this case the ANN takes the form of a standard feedforward ANN.

RL component	Representation
Agent	Ligand
Value/Policy function	Feedforward ANN estimating the Q values
Actions	Two actions: moving forward/backward along one spatial axis
Environment	DockingEnv-v0
States	Mass center of the ligand + Rotational quaternions and the norm
Reward function	Transformed score obtained from Metadock SF. Gradually adds the SF terms according to several conditions

Table 3.2: RL components in the Docking problem.

Moreover, the possible actions for the ligand to be taken are reduced to just two (moving forward/backward along the "x" axis). The states are based on a feature vector that includes the mass center of the ligand plus the rotational quaternions and the norm. Finally, the reward function is computed by adding several partial terms from a given force-field-based SF. These RL components are explained in more detail in the following paragraphs. Likewise, Figure 3.1 has been adapted for this particular implementation, so the operation of QN-Docking can be observed in Figure 3.2. Next, we are going to describe all the RL components and their representation in detail.

As for the actions, at each time-step the agent takes a specific action a_t from the set of possible actions, $A = \{1, \ldots, K\}$. As mentioned earlier, for this problem, two possible actions can be taken by the ligand. In particular, the agent can move forward and backward along the x axis. The idea is to narrow down the search space of Q-values to facilitate the convergence of the algorithm. In the next approaches of molecular encoding, it is intended to include movement and rotation in the three axes. Then, the selected action is used to output the new position or state of the ligand in the environment. For the RL environment ε, the DockingEnv-v0 OpenAI Gym environment is created, including the step, reset, and render methods for the corresponding class in Python.

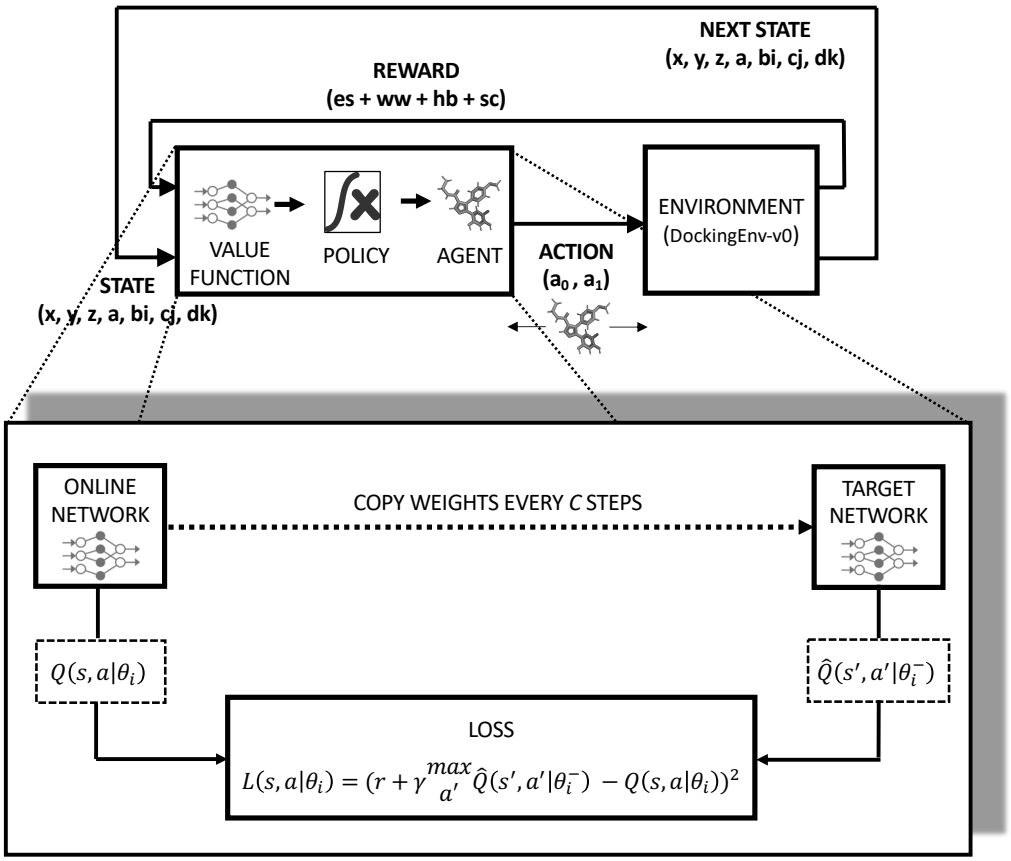

Figure 3.2: Operational schema of QN-Docking. Coordinates (x, y, z) belong to the mass center of the ligand. (a, bi, cj, dk) represents the rotational quaternions and their norm. (a_0, a_1) indicates the possible action to be chosen, that is, moving forward along axis x or backward. Finally, *es*, *ww*, and *hb* stands for the SF terms, which respectively correspond to the electrostatic term, Wan der Waals forces, and hydrogen bonds. In addition, *sc* refers to the overall score from the SF.

With respect to the states, these are vectors $x_t \in R^d$ representing the position of the mass center of the ligand, where t refers to a particular timestep from a given episode and d to the dimension of the states. In addition, it is included the rotational quaternions and their norm in the input data for later extensions of QN-Docking. However, they are not meaningful in this first approach since the ligand is only allowed to move along the "x" axis. Likewise, it should be pointed out that more complete representations of molecules like molecular graphs or 3D grids [66, 173] could be used here. However, those

alternatives were discarded in favor of simplicity to ensure a functional Docking method. For the same reason, information concerning the host is not incorporated in the states nor other kind of knowledge such as atom types or partial charges. That information is indirectly included via the reward function based, in turn, on the SF, which implicitly takes into account the structure of the host. Those two functions–reward and SF–are described in the lines below.

The reward function is one of the most sensitive parts in RL since it serves as a guide for the agent to interact with the environment. It implies a deep knowledge concerning the problem to be solved–in this case, the PLDP problem. The natural choice in this context would be to directly take the raw score from the SF since it represents the quality of the position of the ligand coupled with the host. In particular, the SF used in QN-Docking was originally developed in [128] as part of a blind virtual screening method known as Bindsurf. Such energy function involves the calculation of three major terms, as shown in Equation 3.1: (1) electrostatic interactions; (2) the potential of Lennard-Jones as a mathematical model to solve Van der Waals' forces; and (3) the hydrogen bonds term. In addition, it can include the same solvation term and rotable bonds than AutoDock 4. Algorithm 1 briefly describes how the score is computed by the SF given a particular position of the ligand.

$$\begin{aligned} &\sum_{i=0}^{n}\sum_{j=0}^{m} k\left(\frac{q_i\, q_j}{r_{ij}}\right) + \sum_{i=0}^{n}\sum_{j=0}^{m} 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right) + \\ &\sum_{i=0}^{n}\sum_{j=0}^{m}\left(cos\theta_{ij}\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) + sin\theta_{ij}\, 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right)\right) \end{aligned} \tag{3.1}$$

The score from the SF was tested as the reward signal. However, unlike other settings such as the Atari video-games that DQN was originally designed for, this score is not

Algorithm 1 Sequential baselines for the Lennard-Jones interactions between host and ligand.

for i=1 to $N\_CONFORMATION$ **do**
  for j=1 to N_ATOMS_HOST **do**
    for k=1 to N_ATOMS_LIGAND **do**
      Energy = $4 \times \epsilon \times (term\_raised\_to\_12(j,k) - term\_raised\_to\_6(j,k))$
      Scoring += Energy
    end for
  end for
  S_energy[i] = Scoring
  Scoring = 0
end for

cumulative, it does not increase slightly over time, and it is not always positive. Instead, it is negative most of the time and can drop sharply if (1) the two atoms with positive charge from the ligand and host respectively get too close (electrostatic repulsion); or (2) the ligand overlaps the host (steric repulsion). In fact, the range of the SF goes from stratospheric negative numbers (e.g. -4.5e+21) to 500 at most, depending on the molecules involved.

To show the complexity of the SF, an illustrative experiment was performed. In this experiment, the ligand is pushed forward along the X axis from outside the cyclodextrin toward the center, where the crystallographic solution is, and beyond. The values from the SF are collected and plot against the number of time-steps in Figure 3.3. Although at a first glance the SF may seem relatively simple, a closer look reducing the y-axis minimum limit shows the extremely wide range of its values, going from -713,079,000 to 192. This last maximum value corresponds to the crystallographic spot to be found. Thus, Figure 3.3 shows the large complexity of the Docking SF. Such complexity requires more powerful non-linear functions such as ANNs instead of simpler linear and kernel functions.

As a result, the score from the SF turns out to be too noisy to be employed as a reward signal and it does not favor convergence of the RL algorithm. Alternatively,

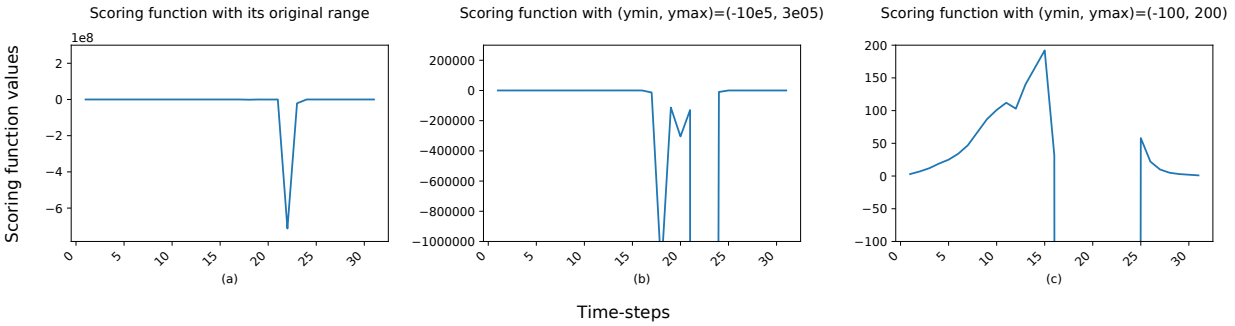

Figure 3.3: Scoring function values for the ligand crossing the center of the cyclodextrin. The y-axis minimum limit is changed to show the extremely wide range of the function due to Van der Waals forces term. In particular, (**a**) y min limit = -1e08 and y max limit = 3e08. (**b**) y min limit = -1e06 and y max limit = 300,000. (**c**) y min limit = -100 and y max axis limit = 200.

other reward signals were tested, including the Euclidean distance between the initial position of the ligand and the crystallographic solution (obviously not realistic since the crystallographic solution is supposedly unknown) and the difference between the energy at the initial and current positions (not valid because it used to get stuck in local optima quite often). Finally, a reward function based on the SF terms seemed to be the most suitable option. These terms are retrieved and gradually added according to several conditions described in Algorithm 2. This reward function enables a more reasonable behavior for the agent. It is worth noting that 15 prototypes for the reward function were tested in total until the final candidate was selected. Most of them are slight modifications or combinations of the aforementioned alternatives. For instance, the Van der Waals forces term was discretized and different partial reward values were assigned to each bucket, several values for the cut-offs were tested, a combination of an Euclidean-distance-based function and a SF-terms-based function was explored as well.

Additionally, the value function in QN-Docking is based on a standard feedforward ANN, also known as dense (fully connected) neural network or multilayer perceptron (MLP). This ANN takes transition tuples containing (s_t, a_t, r_t, s_{t+1})–i.e. the current state, the taken action, the obtained reward, and the next state–as input from the experience

Algorithm 2 Reward function in QN-Docking

Require: *es*: electrostatic term; *ww*: Wan der Waals forces; *hb*: hydrogen bonds; *sc*: overall score from the SF ;λ, ι, δ, ζ: empirically-set cut-offs.
Ensure: reward.
  Initialize reward terms es_r, ww_r, hb_r, and sc_r.
  if *abs(es)* >= λ **then**
    es_r = 1
    if *-ww* >ι **then**
      ww_r = *log(-ww)*
      if *hb* <-1 **and** *abs(ww)* <δ **then**
        hb_r = 1
        if *-sc* >ζ **then**
          sc_r = η
        end if
      end if
    end if
  end if
  reward = add(es_r, ww_r, hb_r, sc_r)

replay dataset. For the task at hand, the optimal number of layers and units for that layer are 1 and 256 respectively, as explained in Section 3.2.2 below. This simpler architecture is chosen over other more compelling alternatives because of the relative simplicity of the inputs/states. In particular, CNNs and RNNs are discarded since QN-Docking does not use 2D/3D grids or molecular sequences as input data, unlike the works analyzed in Section 2.4. Moreover, those memories are sampled from the dataset in minibatches to train the ANN. Specifically, the criterion of absolute Temporal-Difference (TD) error is followed to relatively favor more surprising (better) experiences. In turn, the network outputs the estimated Q values of each action in a given time-step. The action with the highest Q value is selected for the agent at that particular time-step. The specific hyper-parameters of the ANN model are listed in Table 3.3.

3.2.2 Experimental design

Next, the experimental design to evaluate the proposed solution is described. It should be remarked that since this is the first attempt to solve the PLDP problem with RL with

an ANN as function approximator, as far as we know, there are no previous standard experiments to evaluate the proposed method in the Docking context. Moreover, the state-of-the-art Docking methods work differently [109]. They do not start from a given location making subsequent virtual trajectories like QN-Docking. Instead, they normally start from potential hot spots (like alpha carbon atoms) in the host and try different positions following certain heuristics like Monte Carlo or genetic algorithms, for example. Therefore, in addition to setting up the RL environment and the stop conditions, a new experimental setup was created to test QN-Docking.

As for the involved molecules, the evaluation of QN-Docking is based on a beta-cyclodextrin as the target host. These molecules are produced from starch by enzymatic conversion. They stand out for their simplicity and water solubility. As for the ligand, the candidate selected for the experiment is known as kaempferol. Both the target and the ligand were obtained from the Protein Data Bank (PDB).

The conducted experiment entails the training of the agent starting from six different positions independently, as shown in Figure 3.4. Three of these positions fall into the left side of the cyclodextrin set in the origin, while the rest lie on the right side. The agent can move forwards and backwards along the axis that cross through the inner hole of the cyclodextrin and its optimal spot. These six positions on the left and right side of the cyclodextrin are set on purpose to study how the distance to the crystallographic solution affects the convergence of the algorithm. Likewise, the symmetric layout with the origin of coordinates at the center of the cyclodextrin is conscientiously set to analyze the impact of the sign change from the x-axis coordinate on the convergence as well. This mirror set up allowed to discover that the ReLU activation function in the ANN is not suitable for the PLDP problem since the spatial coordinates include negative numbers. The ReLU function transforms the negative number to 0, distorting the original input. In this regard, the hyperbolic tangent is more appropriate.

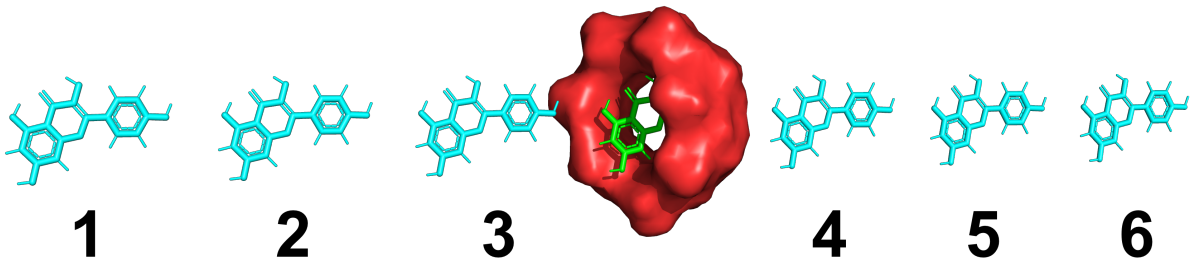

Figure 3.4: Evaluation methodology based on beta-cyclodextrin and kaempferol. The agent is independently trained from six different positions numbered from 1 to 6. The objective for the agent is to discover the optimal solution in the center of the cyclodextrin and stay oscillating around that spot. In a later predictive episode for each of those six training process, the agent is allowed to move according to the learned policy starting in the original position that was trained from and the rest of the five positions, giving rise to a total of 36 runs for the prediction phase.

Additionally, a different maximum number of time-steps per episode is set empirically for each position to guarantee convergence. In particular, for the most distant starting positions (position no. 1 and 6), the agent is trained in episodes with 4,000 maximum time-steps. This limit decreases up to 2,000 steps for intermediate positions (2 and 5) and to 1,000 for closer positions (3 and 4) since the optimal solution is nearer, so the algorithm needs less training in order to converge. After training, the agent is allowed to act according to the learned policy in a new single episode of prediction with a 1,000 time-steps length. In particular, the ligand trained in each starting position is run in the predictive episode starting from that position (just as a step of control to check that it behaves as expected) and the other five. This leads to 36 different runs as a result of crossing the six starting positions in training and prediction. Finally, the goal and desired behaviour for the agent is to check whether it can find the optimal solution, which is known beforehand, and stay put to maximize the reward across the episode.

The experiments were performed on a server with an Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz, 128 GB of RAM, 1 TB SSD Hard Disk. It should be noted that it was

solely conducted on CPU. The massive Single-Instruction Multiple-Thread (SIMT)-based parallelism and the high memory bandwidth make GPUs extremely faster than a CPU in the context of DL. However, for small problems/models as the one at hand, where the number of parameters is not as high as in DL models, CPUs are still considered as more effective and cost efficient. Since the current Q-Network has only 2,562 parameters (without counting on the target neural network) there is simply not much to be gained from faster matrix multiplication. So making use of the GPU would be almost like using a hammer to insert a needle. In fact, the experiment was also conducted with an NVIDIA GeForce GTX 780 GPU but a degradation in the execution speed was observed. Such decrease occurred because the time gain thanks to faster matrix multiplication from the GPU did not balance out the loss due to the data transfer between the CPU and the GPU.

In addition, OpenAI Baselines 0.1.5 [26] was used to deploy QN-Docking. This library is based on Tensorflow and Keras frameworks to design and train ANNs. Specifically, TensorFlow 1.7.0 and Keras 2.1.5 were used for the experiment.

3.2.3 Results and discussion

First of all, a manual hyperparameter tuning [41] focused on execution time is carried out in order to select the optimal combination of RL and ANNs hyperparameters and speed up training. The performed analysis encompasses more than 180 runs with different combinations of the hyperparameters for initial position no. 3. For each hyperparameter, different values are tested. In turn, for each of these values five runs are performed with the intent to reduce the uncertainty caused by the randomness of several elements of the algorithm such as the weights initialization of the ANN or the ϵ-greedy strategy. After performing the five runs, the best value on average is set for the next hyperparameter to be tested. This process of selecting the value of hyperparameters sequentially requires a deep knowledge of the interrelations among them. For example, changing the maxi-

mum of global timesteps of the experiment directly affects the impact of the exploration fraction on execution time. Those interrelations are carefully considered in this analysis. Likewise, this tuning process is cyclically repeated several times until there is no any remarkable improvement in terms of time saving. As a result, execution time is progressively reduced from the original 80 hours to only 12 hours for initial position no. 3.

Figure 3.5 shows the hyperparameters that prove to have a deeper impact in terms of computational efficiency. Overall, exploration fraction and mini-batch size are the most determining ones. With respect to the mini-batch size (Figure 3.5.a), 32 tuples of experiences seems to be the optimal. This value is in line with the original work of DQN. Regarding the neural network architecture, a standard feedforward ANN with up to 5 hidden layers composed of 128 units each is tested (Figure 3.5.b). The ANN with one layer seems to be the most suitable for the task at hand. However, three or more layers make the algorithm unable to converge. Next, it is also tested the number of units for one layer (Figure 3.5.c). Specifically, 256 neurons seem to be the optimal considering the size of the error bars although time saving is not really significant compared to other hyperparameters. The impact of the target network updating frequency C (Figure 3.5.d) is not specially important but values greater than 1 million make the algorithm unfeasible to converge. Moreover, the exploration fraction (Figure 3.5.e) is the most important hyperparameter with respect to execution time. In particular, small values (between 0.005 and 0.02) considerably lower convergence time from more than 50 hours on average (0.1) to just 12. For all these tests, a global maximum limit of 10 million time-steps was set. Finally, Table 3.3 shows the most efficient combination of hyperparameters selected to train the agent after the manual hyperparameter tuning.

Next, it is shown the evolution of the average total reward per time-step during the training process for each initial position in Figure 3.6. It is calculated with respect to the previous 100 episodes. This evaluation metric is chosen among other common

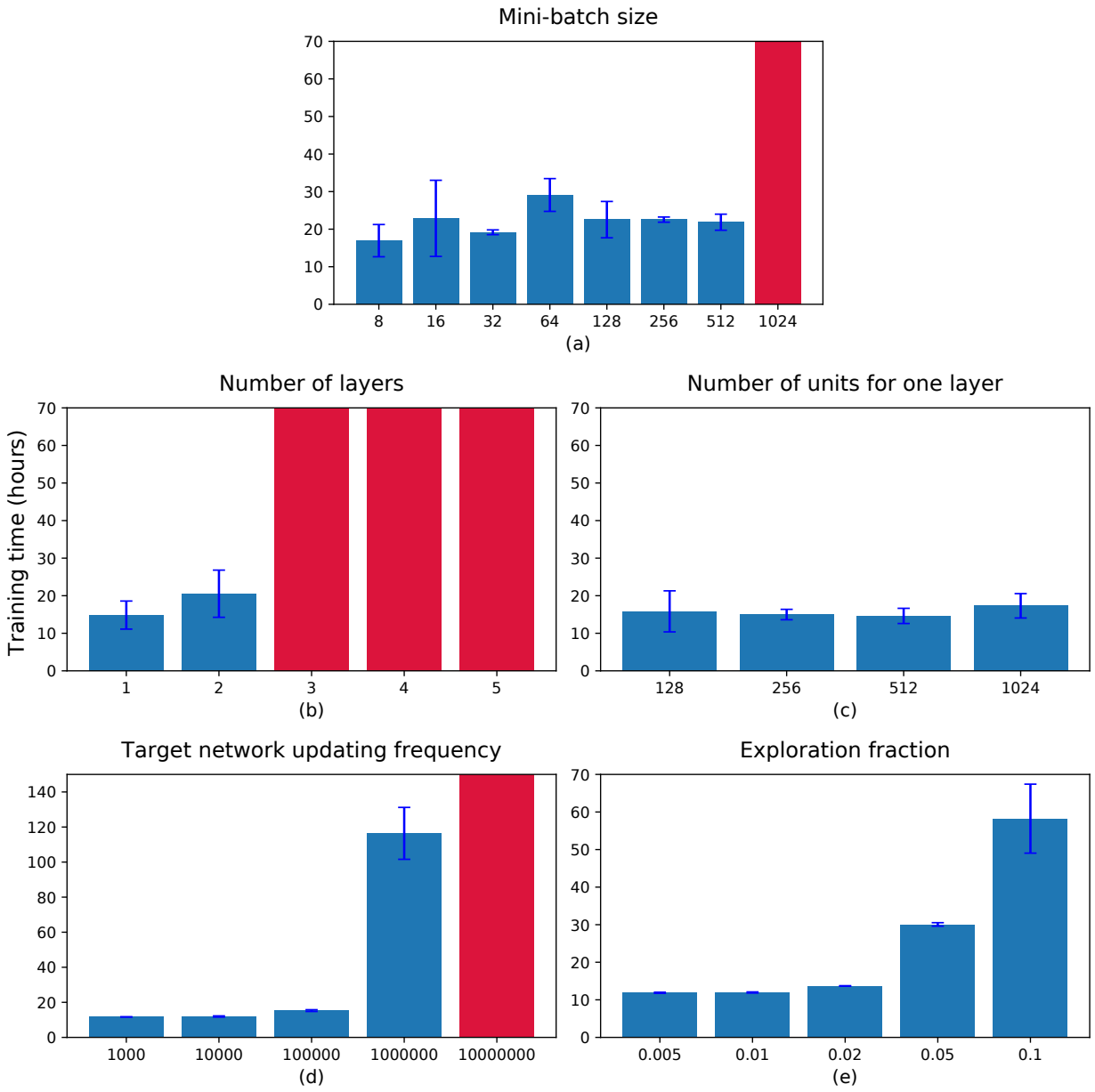

Figure 3.5: Result of hyperparameters analysis of QN-Docking. Hyperparameters values are tested sequentially. Red bars with diagonal stripes indicate that the algorithm failed to converge. Error bars are based on confidence intervals with $\alpha = 0.05$. Note that the y-axis scale is the same for all the bar charts, ranging from 0 to 70, but for 3.5.d, which ranges from 0 to 150.

alternatives in RL like the sum of the cumulative reward, sample complexity, regret, etc. following [100] but note that unlike that study, the current work does not make use of the average Q-values but the average reward itself. For the PLDP problem, the

RL hyperparameters		
Hyperparameter	Value	Description
Number of global time-steps	488,581 / 293,332 / 195,571	Average number of global time-steps completed along the simulation for positions 1&6, 2&5, and 3&4
Global maximum time-steps limit	10,000,000	Maximum time-steps limit along the entire simulation
Maximum time-steps per episode T	4,000 / 2,000 / 1,000	Maximum time-steps limit per episode for positions 1&6, 2&5, and 3&4
State space	7	Real numbers needed to represent a particular state
Action space	2	Real numbers needed to represent the possible actions to be taken by the agent
Shifting length per step	0.1	Ångströms traveled by the ligand in each step when shifting
Rotating angle per step	0.5	Degrees turned by the ligand in each step when rotating
Exploration fraction	0.005	Fraction of entire simulation over which the exploration rate is annealed
ϵ initial value	1	Initial value of ϵ (if ϵ=1, then 100% actions are randomly selected)
ϵ final value	0.02	Final value of ϵ.
γ discount rate	0.99	Discount rate for future rewards
Experience replay pool size N	1,000,000	Number of memories (s_t, a_t, r_{t+1}, s_{t+1}, terminal) to be stored to perform experience replay
Learning start	100,000	Number of initial steps where the agent only takes random actions
Steps C to update target network	1,000	Frequency at which the target network is updated
α PER	0.6	Alpha parameter for prioritized experience replay
β_0 PER	0.4	Initial value of beta for prioritized experience replay
β iterations PER	None	Number of iterations over which beta will be annealed from initial value to 1
ϵ PER	0.000001	Epsilon to add to the TD errors when updating priorities

ANN hyperparameters		
Hyperparameter	Value	Description
Number of hidden layers	1	Number of hidden layers between input and output layers
Hidden layer size	256	Number of units in the hidden layers
Activation function	tanh	Activation function used by hidden units to decide whether they should be activated or not
Update rule	Adam	The parameter update rule used by the optimizer
Learning rate	0.1	Learning rate used by the optimizer
Minibatch size	32	Number of training examples per update

Table 3.3: Values of the hyperparameters used in QN-Docking

average reward is not as noisy as for the Atari environment. Agent in positions 1 and 6 (charts a and f) needs almost 500,000 time-steps and 20 hours of training since these are farthest from the crystallographic solution. Conversely, agent in positions 3 and 4 (c and d) spends less than 200,000 time-steps and 10 hours of training. Thus, the average total reward takes a while until it starts rising. When this happens, it gradually increases until the algorithm converges, suggesting that the agent steadily learns to make better decisions over time. When visualizing its movements in PyMol by the end of training, the ligand tends to stick to the optimal solution oscillating between that position and the next closest one. This is the optimized and desired policy considering that for the agent staying still is not an option.

As for prediction, it is considered to be more suitable a task-oriented metric to evaluate the performance of the proposed method. In particular, the quality of the given position is measure via the Root Mean Square Deviation (RMSD). The heatmap in Figure 3.7a shows the RMSD in ångströms between the last position in the episode of

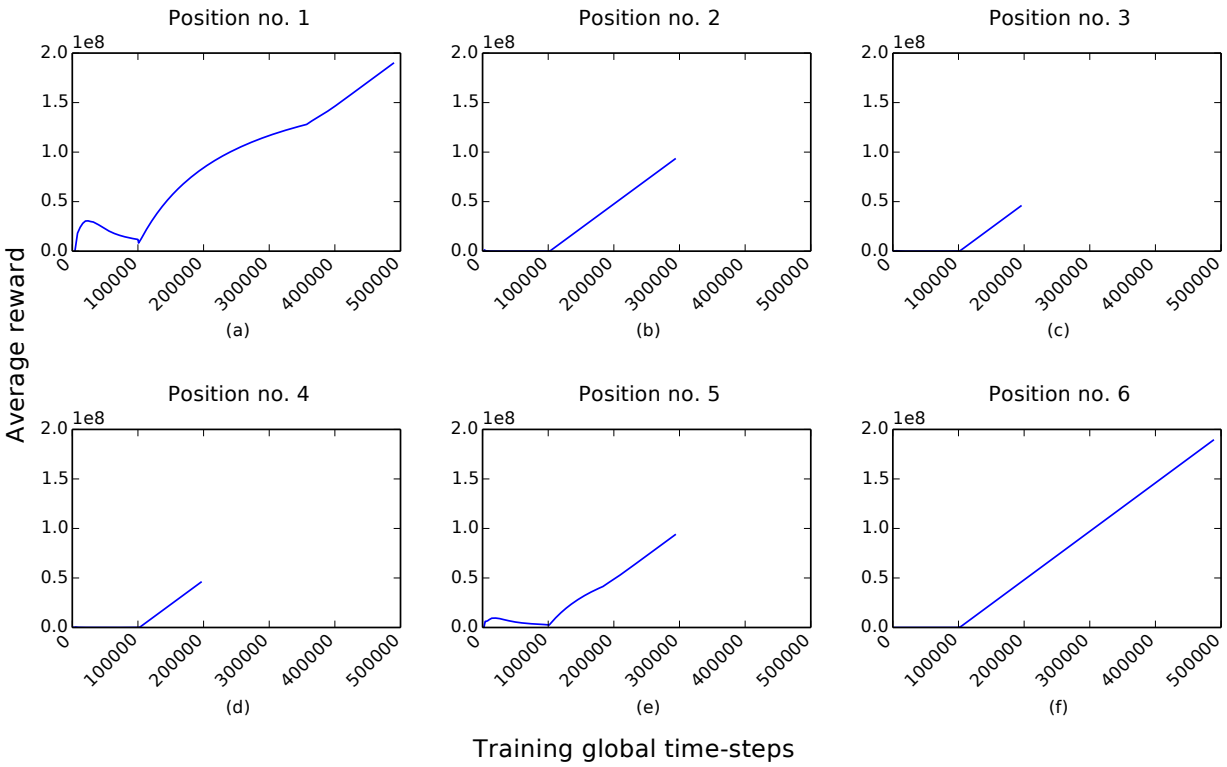

Figure 3.6: Average total reward per time-step during the training process for QN-Docking. The average is calculated considering the previous 100 episodes. For the sake of comparison, both x and y axes share the same range of values in the six charts.

prediction and the optimal solution. This distance is calculated for each pair of training and prediction initial positions. The purpose is to demonstrate that the agent is able to find the optimal spot regardless of which was the initial position during training and, therefore, the validity of the proposed method for the selected ligand-host pair. Specifically, the RMSD varies between 0 and 6 for all cases insinuating that the agent successfully ends up in the optimal location. Consistently with its behavior at the end of the training process, the ligand alternates between the solution (where RMSD = 0) and the next closest position (RMSD = 6) once arrives to the former.

Alternatively, the average RMSD value between the current position at each time-step and the optimal solution is computed across the episode of prediction. The intention of this measure is to ensure that the agent behaves according to the optimal policy. In effect,

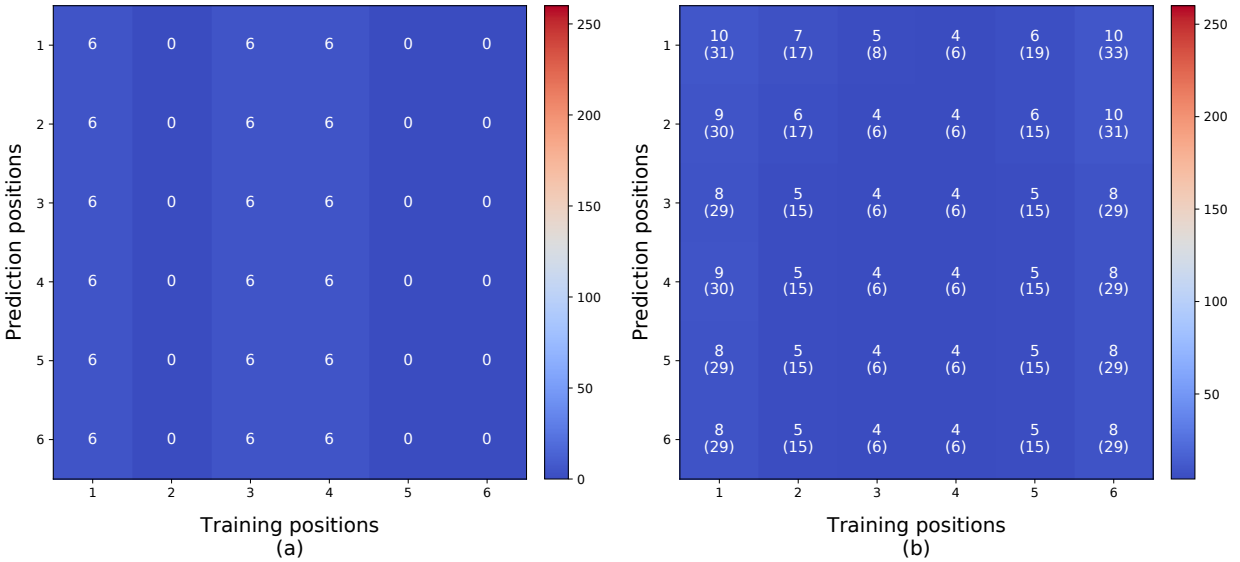

Figure 3.7: (**a**) RMSD between the last position in the episode of prediction and the optimal solution. The distance is computed for each pair of training and prediction initial positions. The maximum distance between the current position and the solution across the 36 pairs is 256 Å. (**b**) Average RMSD between current position and the optimal solution across the episode of prediction. The standard deviation is shown in parentheses.

the average RMSD in Figure 3.7b is pretty low (10 Å as much) for each pair of training and prediction initial positions. This confirms that the ligand does not behave erratically before arriving to the optimal spot. Additionally, the minimum RMSD value between the current position at each time-step and the optimal solution was also calculated. However, the corresponding heatmap is omitted as the RMSD = 0 for all of the 36 pairs of training and prediction initial positions. These results corroborate that the ligand finds the crytallographic solution at least once in every pair, which is coherent with findings in the previous heatmaps.

Furthermore, one could claim that the agent was trained in exactly the same position that it would find in prediction. For the avoidance of doubt, a straightforward experiment is conducted. The agent from position no. 3 is trained with the fastest combination of hyperparameters revealed in the manual hyperparameter tuning but changing the shifting length per step from 0.1 to 0.2 Å. Then, in the prediction phase, the agent is let

QN-Docking		Metadock 2
Training	3451.8	8.250
Prediction	1.033	
Total	3456.774	8.250

Table 3.4: Execution time (seconds) for the pair kaempferol and beta-cyclodextrin

behave according to the learned policy but with a shifting length per step of 0.1 Å. In this way the agent is forced to visit new positions in the prediction phase. Actually, the algorithm converges and the agent goes directly towards the crystallographic solution.

Finally, QN-Docking is compared with a state-of-the-art Docking method in order to prove its validity in terms of efficiency. In particular, METADOCK 2 was chosen opposed to other alternatives such as Autodock 4 and Autodock Vina because the former stands out for being computationally faster [53]. As shown in Table 3.4, our comparison reflects the execution time in seconds for both QN-Docking and METADOCK 2 in the case of kaempferol and beta-cyclodextrin. In particular, QN-Docking starts the episode from position no. 3 and METADOCK 2 follows a heuristic based on a greedy algorithm. As it can be observed, QN-Docking is faster than METADOCK 2 when comparing the prediction stage of the former (1 against 8.2 seconds). However, if the training plus prediction stages are taken into account, METADOCK 2 clearly surpasses QN-Docking (3,456 against 8.2 seconds).

Nevertheless, the training process in RL and supervised learning with ANNs is normally performed a single time. As long as these results could be generalized to other ligand-host pairs, that $8\times$ speed increase would have a tremendous effect on time saving in a VS context where hundreds, millions or even billions [90] of these individual Docking executions are carried out. Figure 3.8 clarifies this last idea by representing the projection of the execution time from both QN-Docking and METADOCK 2. Particularly, this projection has been made for 1,000 Docking executions in a hypothetical VS pipeline.

QN-Docking starts from 3,451 seconds corresponding to the previous training phase while METADOCK 2 starts from zero seconds because such a phase does not exist for traditional heuristics. As the Docking executions increase, METADOCK 2 would start consuming more time than QN-Docking after 478 of those executions. It should be remarked that this line graph has been made for illustrative purposes only. It does not include real data since it is assumed that every Docking execution has the same duration than that of the cyclodextrin. This is obviously not realistic but it is expected for the graph to envisage the potential of QN-Docking.

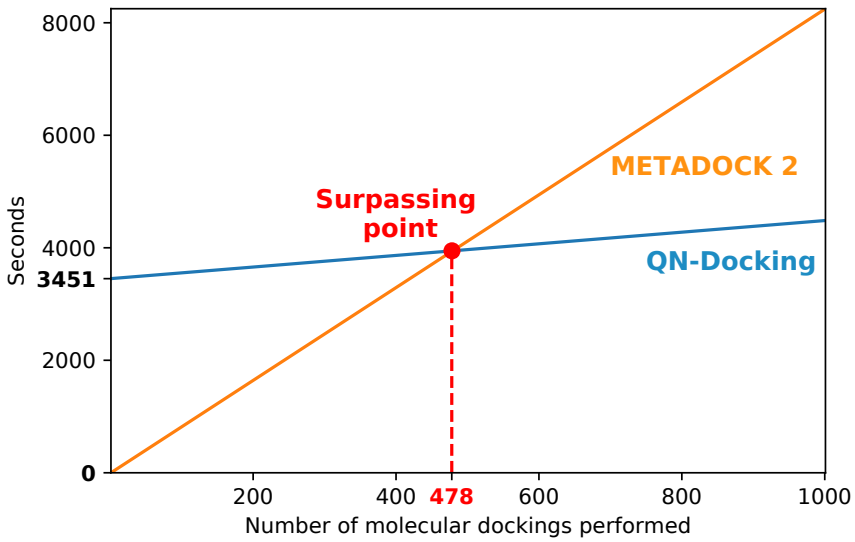

Figure 3.8: Projection of the execution time from QN-Docking and METADOCK 2.

3.2.4 Conclusion

In this first approach, it has been created a system with an embedded Q-Network to train the ligand to look for the optimal solution in a molecular Docking setting by following a reward signal derived from a traditional force-field-based SF. Therefore, the Specific objective 2 has been successfully completed. Results show that, once the agent has been trained, the proposed method is up to 8× faster than a breakthrough Docking software

like METADOCK 2. Of course, if the training time is also considered, the total execution time of QN-Docking is not shorter than METADOCK 2. However, as in every problem of RL and supervised learning with ANNs, the training process is typically performed only once. If the proposed method could be generalized to other ligand-host pairs, its impact on time saving would be enormous in VS workflows where hundreds, millions or even billions of these Docking executions are carried out. Therefore, QN-Docking opens a fresh, promising opportunity to develop a general and faster Docking method in order to contribute to accelerate drug discovery and be able to deliver medicines to patients in a shorter time frame.

3.3 Molecular encoding based on images

In this second approach, it is plan to reach the Specific objective 3 stated in Section 1.2. In other words, to implement the basic system created in the Specific objective 1 by representing the states through images to solve the Protein-Ligand Docking Prediction problem in a simplified environment. To do so, it is assumed the secondary hypothesis, which states that it is necessary to add structural information of both the ligand and the receptor to correctly inform the Deep Q-Network to solve the problem. As previously shown in Section 2.3, there are several alternatives for molecular encoding but none of them can be considered as a clear winner. Consequently, an innovative decision is made by selecting images from 2D drawings of molecules to represent the Docking scene. Additionally, a multi-view CNN is developed to avoid molecular overlapping in those images. The resulting system is termed Multi-View Deep Q-Network (MVDQN). It is tested on a setting with similar conditions to that of QN-Docking in order to evaluate the new system before moving to a more complicated, ambitious scenario. In this case, MVDQN successfully achieves similar results than the implementation based on

RL component	Representation
Agent	Ligand
Value/Policy function	Multi-View Convolutional Neural Network
Actions	Two actions: moving forward/backward along one spatial axis
Environment	DockingEnv-v1
States	1 image of the Docking scene
Reward function	Transformed score obtained from Metadock SF. Gradually adds the SF terms according to several conditions. A penalty for ending the episode sooner is added

Table 3.5: RL components in the Docking problem based on images to represent the states.

feature vectors with the kaempferol and the cyclodextrin during the training phase. In prediction, however, results are mixed. As a result, the Specific objective 3 is partially attained.

3.3.1 Specific methodology

The reader is referred again to Section 3.1 to be able to wholly comprehend the aspects contained in this section. Next, Table 3.1 is updated with the peculiarities of the RL components of MVDQN and listed in Table 3.5. The selected architecture for the ANN in this case is a CNN considering that the input data is based on images. The action space includes the previous 2 movements. DockingEnv-v1 was built with OpenAI Gym for the RL environment including the corresponding modifications in the step, reset, and render methods within the Python class. As for the states, these are built upon the values corresponding to the pixels from 2D drawings of molecules in gray scale. In this experiment, only one image of the Docking scene is render. The reward function is also slightly modified. Its changes along with those of the rest of the RL components are described more in depth in the lines below. Likewise, Figure 3.9 summarizes the operation of MVDQN.

With the respect to the action space, the two actions from QN-Docking are included

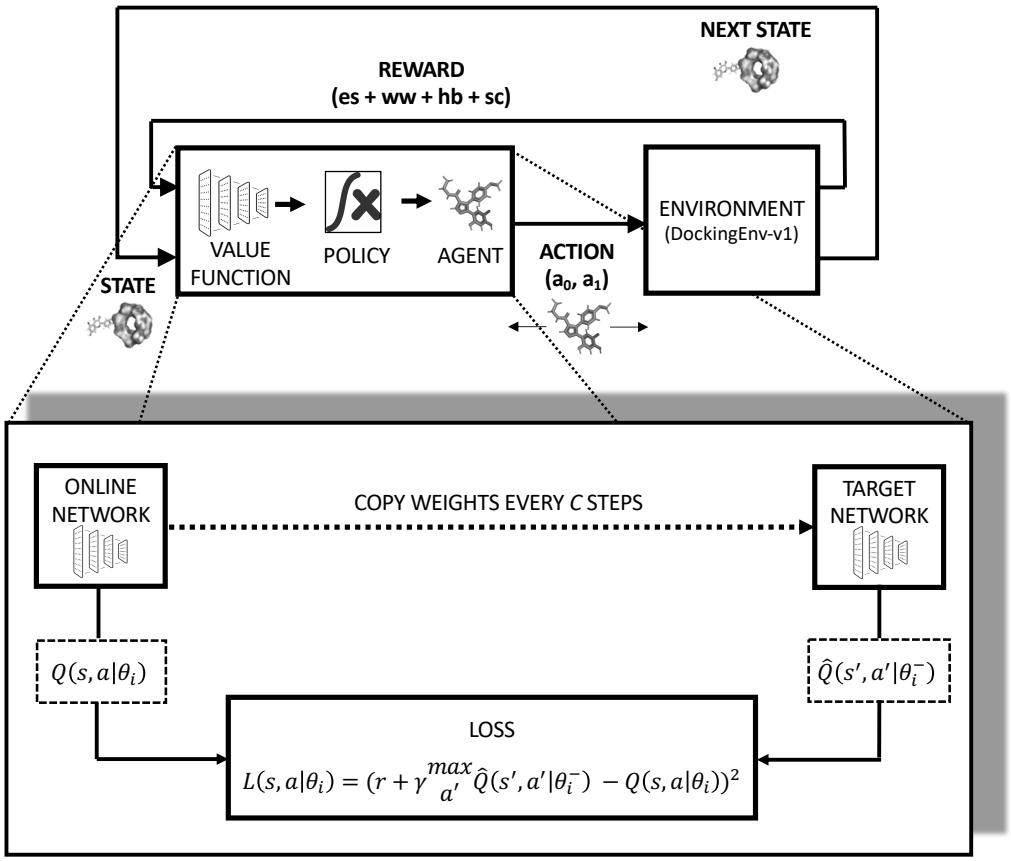

Figure 3.9: Operational schema of MVDQN. States are represented by pixels of an image from the Docking scene (although it can accept more than one perspective). Possible actions correspond to moving forwards and backwards along the "x" axis (translation in the three axes and rotations can be activated as well). The value function is based on a Multi-View Convolutional Neural Network. The reward function present minor changes, as explained in Algorithm 3. The rest of the RL components remain unaltered with respect to the previous approach represented in Figure 3.2.

(moving forward/backward along the x axis) in this second approach. Shifting in the three axes and rotation could have been added as well, but it is preferred to deal with one problem at a time and to not complicate the scenario in excess to avoid undesirable pitfalls. The states now consist of images based on 2D drawings of molecules that represent the Docking scene. As previously shown in Section 2.3, none of the alternatives in Docking for molecular encoding are the panacea. Thus, a pioneering approach is

proposed in this section by selecting images to represent the molecules involved in the ligand-protein interactions. This alternative way of representing molecules has been used before in drug discovery Goh et al. [36, 37], Matsuzaka and Uesawa [94], Rifaioglu et al. [123] but not in the protein-ligand interaction prediction context. We believe that images could be a more suitable input to compress pertinent information related to Docking while harnessing the power of more complex ANNs in this context beyond standard feedforward ANNs.

In particular, the images are taken using the Python's API of PyMol and saved as Portable Network Graphics (PNG) files. These are gray-scale images with a resolution of 84 by 84 pixels. Furthermore, the optimal position for the ligand is often situated in a deep cavity in the receptor. Therefore, overlapping between the ligand and the receptor in the 2D drawings is something usual when searching the optimal location around the surface of the receptor. To overcome this hurdle, it is proposed to use a Multi-View CNN [148]. Although this multi-view scheme is developed to extend the basic system referred in the Specific objective 1, it should be noted that only one image of the Docking scene is used in the experiment performed in this section. More details about MVDQN using more than one view is discussed in Section 4.1.

There are also changes with respect to the reward function. First, the SF from METADOCK/Bindsurf was translated from C to Python in order to save execution time. Generally speaking, C programs run faster than those written in Python because it is a compiled programming language, as opposed to an interpreted language like Python. Nevertheless, the way METADOCK is designed involves several reads and writes of the scoring and molecular files to disk, which ends up increasing the execution time excessively. Second, the reward function was slightly improved to provide a more clear reward signal and facilitate convergence (see Algorithm 3). Particularly, a penalty for the agent is imposed equal to the maximum reward that it can obtained if it finds the optimal

solution (η). The purpose of this penalty is to strengthen the prevention of the ligand for finishing the episode earlier than it should according to the maximum timesteps per episode set at the beginning of the simulation. Finally, the used of the energy function from AutoDock Vina is also explored. Unlike the energy function from Bindsurf, which is a force-field-based function, the SF from AutoDock Vina is a hybrid scoring function: empirical and knowledge-based function (see Section 2.1). It is inspired in the X-Score function [164], mainly differing in some terms and in the parametrization method. One of the strongest points of this SF is that is commonly used as a baseline when comparing diverse energy functions. Nevertheless, its use is not feasible in the proposed system due to its high computation cost through the Python's API, as explained in the lines below.

Algorithm 3 Reward function in QN-Docking

Require: *done*: whether the current episode has ended; *current_step*: present timestep; *max_timesteps_per_episode*: maximum number of timesteps per episode; *es*: electrostatic term; *ww*: Wan der Waals forces; *hb*: hydrogen bonds; *sc*: overall score from the SF ;λ, ι, δ, ζ: empirically-set cut-offs.
Ensure: reward.
  Initialize reward terms es_r, ww_r, hb_r, and sc_r.
  if *done* = *True* **and** *current_step* <*max_timesteps_per_episode* **then**
    reward = $-\eta$
  else
    if *abs*(*es*) >= λ **then**
      $es_r = 1$
      if -*ww* >ι **then**
        $ww_r = 1$
        if *hb* <0 **then**
          $hb_r = 1$
          if -*sc* $\geq \zeta$ **then**
            $sc_r = \eta$
          end if
        end if
      end if
    end if
  end if
  reward = add(es_r, ww_r, hb_r, sc_r)

As for the ANN architecture, the value function in QN-Docking relies on a modified version of the Multi-View CNN [148]. This ANN was originally conceived to recognize

3D shapes according to view-based descriptors instead of native 3D formats, such as voxel grid or polygon mesh. The authors make use of a CNN architecture that combines information from multiple views of a 3D shape into a single and compact shape descriptor offering even better recognition performance than state-of-the-art 3D shape descriptors. More specifically, the architecture includes a basic CNN to handle each view separately. Then, a global view-pooling layer is attached after the previous convolution layers. In this view-pooling layer an element-wise maximum operation is performed across the views. Finally, a unique CNN is use to solve the class prediction problem.

Several works have improved the results obtained by Su et al. [148], like Leng et al. [77], Qi et al. [116], or Kanezaki et al. [65]. This last approach leads the Modelnet40 leaderboard at the time this thesis is being written [172]. Namely, it achieves a classification accuracy of 97.37% while Su et al. [148] gets 90.1%. These superior results come at the expense of more complexity, problem specialization, and the use of additional tricks. For example, previous approaches in 3D shapes recognition are based on known viewpoint labels for training, while the method in Kanezaki et al. [65] treats the viewpoint labels as latent variables learned in an unsupervised manner during the training using an unaligned object dataset. Thus, the architecture in Su et al. [148] is selected to be applied to the Docking context due to its higher simplicity and generalizability compared to more advanced algorithms such as Kanezaki et al. [65] or Zhang et al. [175].

Figure 3.10 shows the architecture of the neural network in MVDQN, where it is combined a Multi-view CNN with a dueling architecture [167]. In particular, three convolution layers handle the input data (a PyTorch tensor with shape $(batchsize, no.of views, inputheight, inputwidth, no.of channels)$). The filter size for each of these layers is 8, 4, and 3, respectively, combined with a stride of 4, 2, and 1. As the reader may have noticed, there is no max nor average pooling layers after each convolutional operation. Pooling layers in general address the problem known as local trans-

lation invariance—i.e. the fact that the output feature maps from the convolution layers are sensitive to the location of the features. One approach to deal with this sensitivity is to down sample the feature maps. This makes the resulting down sampled feature maps more robust to changes in the position of the feature in the image. However, it was preferred to respect the original layout devised by Mnih et al. [100] where any pooling layer was attached. The reason behind this decision may be to avoid distance distortion in the images from the Atari 2600 emulator. In this regard, distance is also crucial to compute the binding affinity for a given ligand-receptor pair.

After the corresponding matrix multiplication, the global view-pooling layer performs an element-wise maximum operation across the three views. The output is then split into two streams following a dueling architecture. The state-value stream helps to distinguish which states are good or bad per se, without taking any particular action. The action advantage stream, however, focuses on identifying the best actions for each state. Both streams are composed of two consecutive fully connected layers. The first of these layers has an output size of 512 units in both streams. In the second one, however, the output size is equal to one in the case of the state-value stream, while for the action advantage stream it corresponds to the number of actions. As in the previous approach based on the simplified feature vector, the final output consists of the estimates of the Q values indicating the quality in the long run of each action in each state. There are no changes with respect to the application of the prioritized experience replay during training. Finally, the specific hyperparameters of the ANN model are listed in Table 3.6.

3.3.2 Experimental design

MVDQN is tested using the previous setting based on the kaempferol and beta-cyclodextrin. This experiment is performed for sake of consistency, to ensure that the new approach is able to attain at least similar results when solving the PLDP problem.

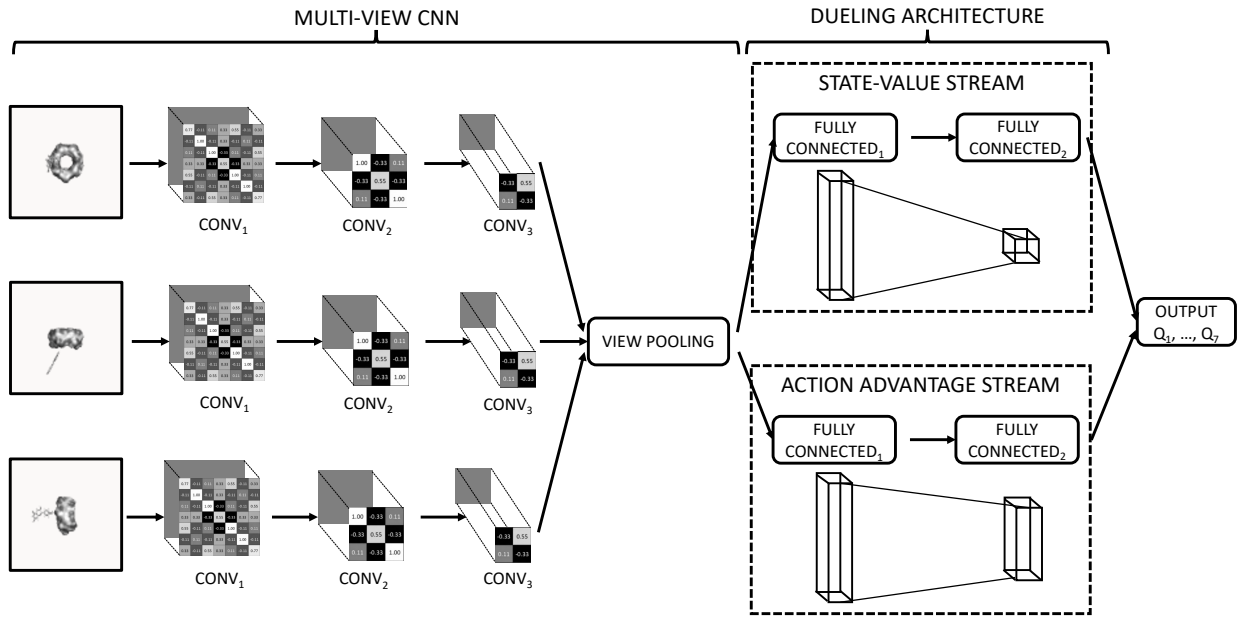

Figure 3.10: Example of the neural network architecture of MVDQN for three views.

As for hardware, the experiment was performed on a server with an Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz, 384 GB of RAM, 32 TB SSD Hard Disk. Unlike the feature-vector-based approach, the execution relied not only on the CPU but on the GPU as well. It should be recalled that the molecular encoding is now based on images and the ANN is a Multi-View CNN. As a consequence, the total number of trainable parameters amounts to 3,285,667 for 2 actions and 1 view, which in both cases is considerably more than the 2,562 from the previous approach of QN-Docking. This makes the use of the GPU essential to speed up the training process. Specifically, the server is equipped with two Nvidia GeForce RTX 2080 Ti.

With respect to software, a vanilla version of the DQN algorithm was adapted from Cook [18]. This kind of simpler implementations offers more control and flexibility to adjust the code to other contexts, as it is the case. The code for the Prioritized Experience Replay was borrowed from Silveria [142]. This implementation is based on

an unsorted sum tree model to sample the tuples from the replay buffer more efficiently. MVDQN was coded using PyTorch 1.2.0 instead of relying on the combo of Tensorflow plus Keras. The change of framework is due to PyTorch's higher ease of use at the time the application was built. Namely, PyTorch works with dynamic graphs, so one does not need to statically define it first to run the code and start debugging it. This has changed in the past year, however, with the development of TensorFlow 2.x. In addition, OpenAI Gym 0.17.2 was used to create the RL environment of DockingEnv-v1. PyMol 2.5.0 was chosen to render the images from the Docking scene, and move and rotate the ligand in the three-dimensional space. Finally, AutoDock Vina 1.2.2 was employed to offer an alternative SF beyond that of Bindsurf.

3.3.3 Results and discussion

The following lines are focused on a similar scenario to the previous feature-vector-based approach to dock the kaempferol and beta-cyclodextrin. But in this case, MVDQN is tested. Analogously, a manual hyperparameter tuning centered on execution time is conducted although a little less exhaustively. In particular, the performed analysis encompasses more than 40 runs with different combinations of the hyperparameters for initial position no. 3. Like in QN-Docking, different values are tested for each hyperparameter. In this case, each value is tested three times with different seeds and the average time is then calculated. After performing the three runs, the best value on average for that particular hyperparameter is set for the next hyperparameter to be tested following a cascading pattern.

Before analyzing the results, it is worth to highlight the slight differences in the implementation of MVDQN and that of QN-Docking. As one might expect, the definition of the hyperparameters related to the DQN algorithm itself barely varies compared to the previous section. One remarkable difference is that in the implementation of OpenAI

Baselines, the ϵ value within the ϵ-greedy strategy is annealed by specifying a fraction of the total number of timesteps set for the whole simulation through linear interpolation. In this case, epsilon decay is used. This method consists of a multiplicative factor (<1) that decreases the value of ϵ in each episode, not in each timestep like in linear interpolation. Another minor change in the new implementation affects the hyperparameter that takes over the soft updating of the target network. Instead of updating the target network every C steps, a τ hyperparameter following the equation $\theta_{target} = \tau \times \theta_{local} + (1 - \tau) \times \theta_{target}$ is applied. Thus, if $\tau = 1e^{-4}$, the target network will update every 1,000 steps; if $\tau = 1e^{-3}$, it will update every 10,000 steps; etc. So, this replacement of the C steps by τ is equivalent and does not entail a real change in the algorithm compared with the previous approach but it is necessary to understand its meaning. The definition of the rest of the hyperparameters like the exploration fraction, γ discount rate, learning start, etc. remain as they are.

As for the hyperparameters related to the Docking problem (maximum time-steps per episode, shifting length per step, rotating angle per step, etc.), there are no changes but in the state space. In the previous system, the states are based on a feature vector that includes the mass center of the ligand plus the rotational quaternions and the norm, so the state space is equal to 7. In this proposal, the state space is equal to $no.\ of\ views \times input\ height \times input\ width \times no.\ of\ channels = 1 \times 84 \times 84 \times 1 = 7,056$. The action space remains equal to 2 (moving forwards and backwards along the x axis). Likewise, there are new hyperparameters related to the Multi-View CNN: the height of the image resolution, width, number of channels, and the number of views or perspectives. As mentioned before, the number of convolutional layers remains fixed with respect to the architecture from the original algorithm of DQN, so it is not considered a hyperparameter. The hyperparameters of the CNN such as the kernel size, stride, or padding are not tweaked either. The same happens with the number of fully-

connected layers in the block of the dueling architecture. The definition of the rest of the hyperparameters like the update rule, learning rate, or the minibatch size remain unchanged.

Figure 3.11 shows the most common and important hyperparameters in DL and/or those that prove to have a deeper impact in terms of computational efficiency. In the case of the initial position no. 3, the execution time is gradually decreased from the initial average of 5 hours (corresponding to the best QN-Docking's hyperparameter combination applied to MVDQN) to 3 hours and 54 minutes. As expected, this reduction in the execution time is not as pronounced as in the hyperparameter analysis performed in QN-Docking since the initial combination is precisely that of QN-Docking, so the room for improvement is narrower. But it is worth to conduct such analysis as a control measure.

Thus, a learning rate of 0.01 turns out to be the best value, as in QN-Docking. It should be noted that values equal or greater than 0.1 make the algorithm unable to converge. With respect to the frequency to update the target network (τ), $\tau = 1e^{-3}$ (or equivalently 10,000 steps) is faster for convergence than the value of $\tau = 1e^{-4}$ (1,000 steps) from QN-Docking. This configuration saves around 1 hour of computation. However, it is observed that MVDQN is not able to converge during training with that value ($\tau = 1e^{-3}$) if the ligand starts from farther positions. Therefore, $\tau = 1e^{-4}$ is the final value selected to continue with the analysis. Surprisingly, the mini-batch size, traditionally one of the hyperparameters with greater impact in training performance in DL, barely affects the execution time but for very large values (>1,024). Therefore, the original value for the batch size in DQN (32) is maintain for the next hyperparemeter exploration.

Finally, the activation function proves to be the most decisive hyperparameter in this analysis. The best activation function in QN-Docking is the hyperbolic tangent (TanH). However, in MVDQN nor the TanH nor sigmoid functions enable convergence. This

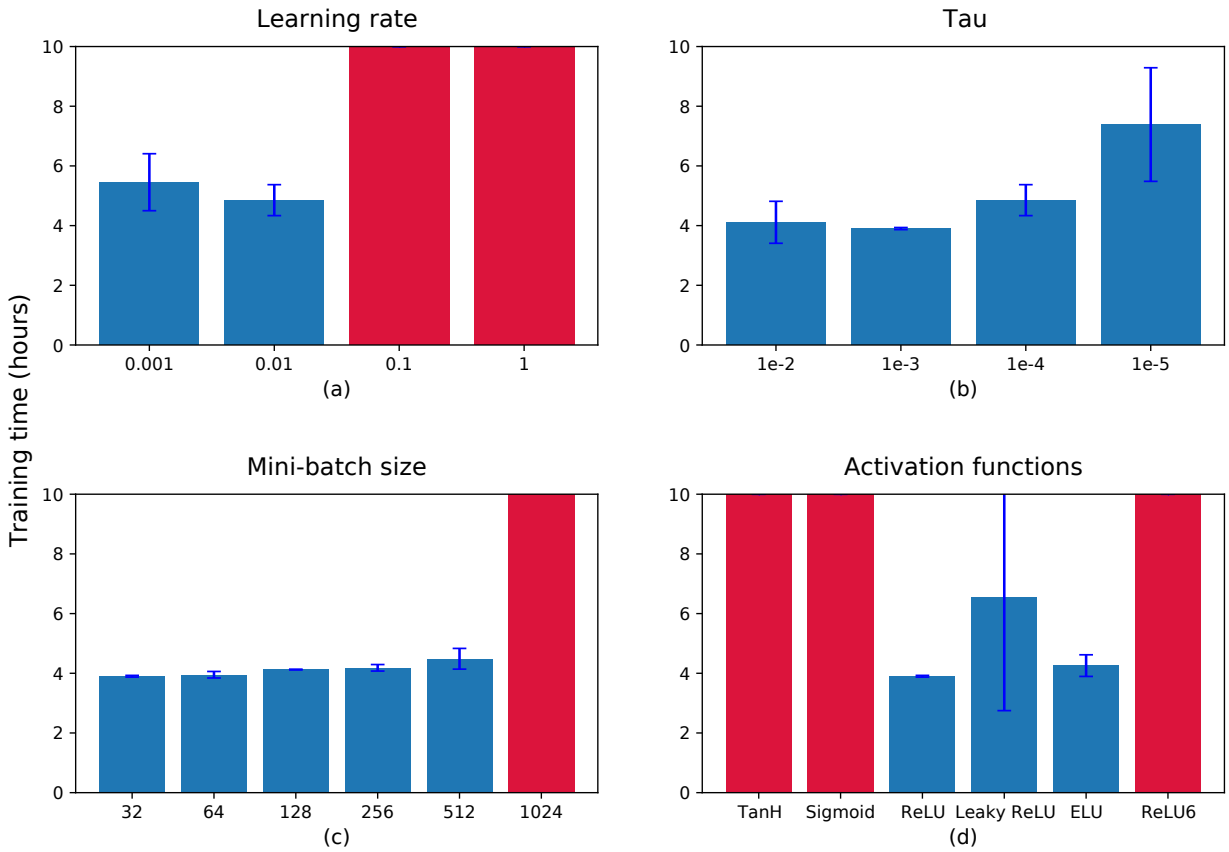

Figure 3.11: Result of hyperparameters analysis of MVDQN. Hyperparameters values are tested sequentially. Red bars with diagonal stripes indicate that the algorithm failed to converge. Error bars are based on confidence intervals with $\alpha = 0.05$.

is probably due to their tendency to saturate [41]: large values snap to 1.0 and small values to -1 or 0 for TanH and sigmoid, respectively. Once saturated, it is quite hard for the learning algorithm to keep adjusting the weights to improve the performance of the model. Goodfellow et al. [41] also points out another flaw of these two traditional activation functions. In large networks, they tend to suffer from the vanishing gradient problem because the deeper layers fail to receive useful gradient information, so the error is back propagated through the network and used to update the weights. Nonetheless, this not the case of MVDQN since their CNNs only have 3 convolutional layers. The activation functions that do converge are the Rectified Linear Unit (ReLU) and some of its more popular variants. In particular, the standard ReLU is the fastest activation

RL hyperparameters		
Hyperparameter	Value	Description
Number of global time-steps	152,855 / 92,105	Average number of global time-steps completed along the simulation for positions 2&5, and 3&4
Global maximum time-steps limit	10,000,000	Maximum time-steps limit along the entire simulation
Maximum time-steps per episode T	2,000 / 1,000	Maximum time-steps limit per episode for positions 2&5 and 3&4
State space	7,056	Real numbers needed to represent a particular state
Action space	2	Real numbers needed to represent the possible actions to be taken by the agent
Shifting length per step	0.1	Ångströms traveled by the ligand in each step when shifting
Rotating angle per step	0.5	Degrees turned by the ligand in each step when rotating
ϵ initial value	1	Initial value of ϵ (if ϵ=1, then 100% actions are randomly selected)
ϵ final value	0.02	Final value of ϵ.
ϵ decay	0.99	Multiplicative factor per episode for gradually decreasing ϵ.
γ discount rate	0.99	Discount rate for future rewards
Experience replay pool size N	50,000	Number of memories (s_t, a_t, r_{t+1}, s_{t+1}, terminal) to be stored to perform experience replay
Learning start	50,000	Number of initial steps where the agent only takes random actions
τ to soft update target network	$1e^{-4}$	Indirectly controls the frequency at which the target network is updated
α PER	0.6	Alpha parameter for prioritized experience replay
β_0 PER	0.4	Initial value of beta for prioritized experience replay
β iterations PER	None	Number of iterations over which beta will be annealed from initial value to 1
ϵ PER	0.000001	Epsilon to add to the TD errors when updating priorities

ANN hyperparameters		
Hyperparameter	Value	Description
Image height	84	Height regarding image resolution
Image width	84	Width regarding image resolution
Number of channels	1	Number of filter maps applied in each convolution layer
Number of views	1	Number of perspectives from which the images of the Docking scene are taken
Activation function	ReLU	Activation function used by hidden units to decide whether they should be activated or not
Update rule	Adam	The parameter update rule used by the optimizer
Learning rate	0.01	Learning rate used by the optimizer
Minibatch size	32	Number of training examples per update

Table 3.6: Values of the hyperparameters in MVDQN

function followed by the Exponential Linear Unit (ELU). Leaky ReLU, the variation with a small slope for negative values instead of altogether zero, is also able to converge although slightly slower. Finally, Table 3.6 shows the most efficient combination of hyperparameters selected to train the agent after the manual hyperparameter tuning.

In Figure 3.12, it is shown the evolution of the average total reward per time-step during the training process. The average reward is also computed considering the previous 100 episodes. In this case, however, it is considered 4 positions instead of 6. Position 1 and 6 (see Figure 3.4) are discarded as unnecessary since they are deemed unrealistic for being too far from the receptor. This is because most of Docking methods start from potential hot spots like alpha carbon atoms, not from locations far from the surface of the receptor. Thus, position no. 2 is 3 angstroms to the left of the optimal solution, position no. 3 is 1.5 Å, position no. 4 is 1.5 Å to the right, and position no.

5 is 3 Å also to the right. Thus, the maximum distance between both end positions (2 and 5) is 6 Å in total although the ligand is allowed to move 1 Å beyond those opposite sides. Agents in positions 2 and 5 (charts a and d) needs more than 140,000 time-steps and 8 hours of training since these are farthest from the crystallographic solution. In positions 3 and 4 (c and d), however, it spends less than 100,000 time-steps and 6 hours of training. As in Figure 3.6, the average total reward gradually increases until the algorithm converges, suggesting that the agent steadily learns to make better decisions over time. When visualizing its movements in PyMol by the end of training, the behavior of the agent is similar to the one in QN-Docking i.e. it tends to stick to the optimal solution.

Finally, the prediction or inference phase is tested analogously with respect to QN-Docking. In other words, the agent is allowed to act starting from positions 2, 3, 4, and 5 according to the learned weights in each of those positions during training. This gives rise to $4 \times 4 = 16$ combinations based on the available prediction (starting) and training positions. In each of them, the RMSD is also analyzed to deem the quality of the learning acquired at training. Figure 3.13a shows the RMSD in angstroms between the last position in the episode of prediction and the crystallographic solution. The distance is calculated for each pair of training and prediction positions, oscillating between 0 and 5 angstroms at most. Those pairs whose RMSD is larger than 0 mean that the agent does not end the episode of prediction in the crystallographic solution but somewhere else. As expected, those episodes where the starting and training positions match (i.e. the diagonal of the heatmap) behave optimally. In addition, there are some others combinations of prediction and training positions outside the diagonal where the ligand also finds the solution and remains in it. However, the ligand does not lie in the optimal spot in most of those combinations outside of the diagonal. This is confirmed in Figure 3.13b through the average RMSD between the current position of the ligand and the optimal solution

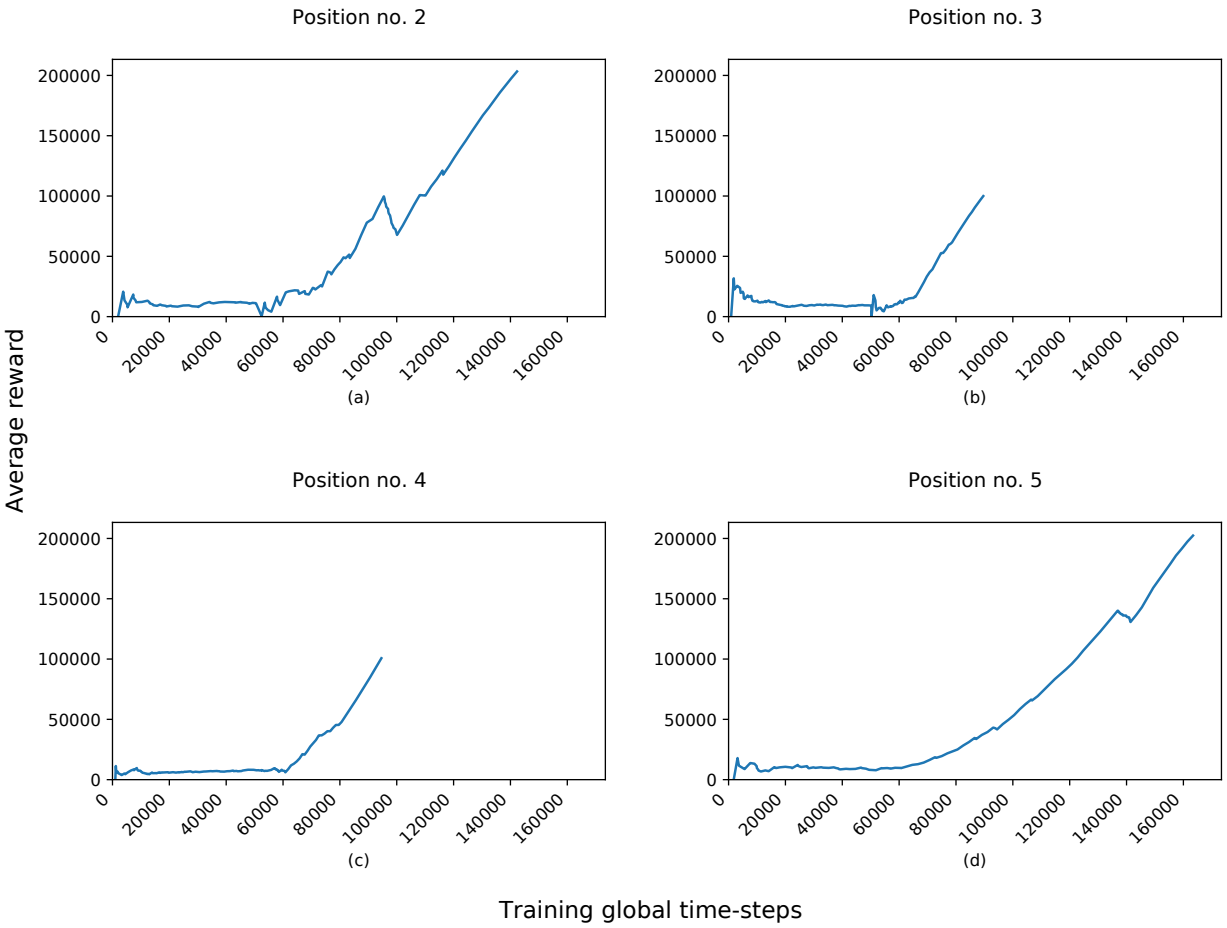

Figure 3.12: Average total reward per time-step during the training process for MVDQN for positions 2, 3, 4, and 5 (see Figure 3.4). The average is calculated considering the previous 100 episodes. For the sake of comparison, both x and y axes share the same range of values in the four charts.

across the episode of prediction. For example, for the ligand trained in position no. 2 and starting from position no. 5, the average distance across the whole episode is 4.03 angstroms, which proves that the ligand is not able to traverse from one end to the other.

The sub-optimal behavior in those combinations whose average RMSD across the episode of prediction is larger than 1 was confirmed by visualizing the agent in motion with PyMol. In some of those cases, the agent tends to stick around ghost positions somewhere in the middle of the one-dimensional path of the "x" axis. In some others, the agent drifts away beyond the boundary in one of the ends of such path. This suggests

that the model could suffer from overfitting [41], trying to move the agent to fix positions in the space. In other words, the Multi-View CNN has learned the input data excessively well, performing poorly on hold out samples (i.e. new starting positions). Overfitting, in general, can be addressed by adding more training examples and/or reducing model complexity. The first measure could be applied by including more perspectives of the Docking scene in the input data for the same ligand-receptor pair. Another alternative would be to change the molecular pair involved in the Docking process to get more examples. But this would entail changing to a new environment, so it is not a real solution. In addition to adding more views, it could be tried to reduce the capacity of the ANN by applying a regularization methods such as dropout.

Another reasonable explanation for such defective behavior is related to molecular encoding. Perhaps the current representation of the Docking scene is not enough for the model to approximate the optimal policy function. Again, adding more perspectives of the scene could shed a bit of more light on the neural network in order to generalize to new starting positions later in the prediction phase. In addition, we believe that it would be eventually necessary to add more chemical knowledge relevant for Docking in the states of the algorithm beyond structural information. For example, the number of channels could be increased to work with color images indicating the atom type. Moreover, those channels could be expanded to contain information about partial charges, pharmacophoric properties, atom connections, hybridization, aromaticity, amino acid types, etc. Finally, the resolution of images could be increased for the model to be able to distinguished certain positions of the ligand more clearly, even though it could entail many more parameters to train at the same time.

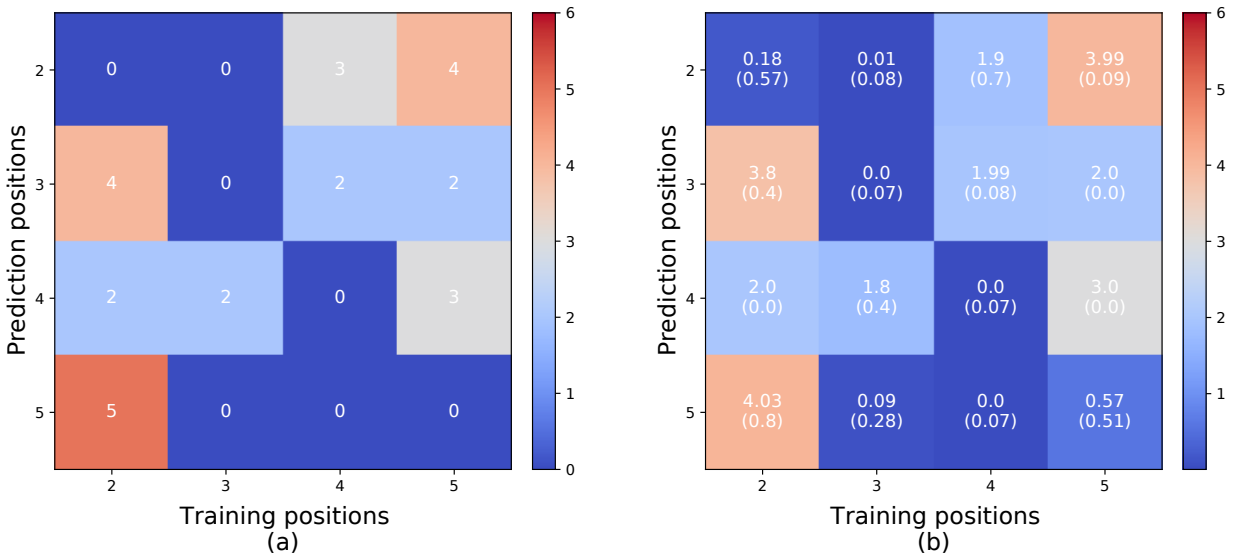

Figure 3.13: (**a**) RMSD between the last position in the episode of prediction and the optimal solution. The distance is computed for each pair of training and prediction initial positions. The maximum distance between the current position and the solution across the 16 pairs is 60 Å. (**b**) Average RMSD between current position and the optimal solution across the episode of prediction. The standard deviation is shown in parentheses.

3.3.4 Conclusion

In this second proposal of implementation, similar results to QN-Docking were obtained in the context of the kaempferol and beta-cyclodextrin during the training phase. Specifically, it is able to steadily learn to approach the ligand to the crystallographic solution and stick to it until the episode finishes. In the prediction phase, the agent behaves optimally if it uses the weights learned from the same position where it starts. In addition, there are some other combinations of training and starting positions where the agent is able to find the solution. But in most of those combinations the agent acts suboptimally by sticking around ghost positions more or less far from the solution or by drifting away beyond the physical boundaries of the environment. These problems could be faced by applying solutions to a potential overfitting problem or improving molecular encoding for the RL states. Thus, this scenario seems promising because there is much room for

improvement. In other words, the system could be able to find the optimal solution in relative small receptors and limited action spaces with further adjustments. In conclusion, the Specific objective 3 has been partially achieved. In fact, a manuscript based on these last findings has been prepared and submitted to the editors of The Journal of Supercomputing (Q2). We are currently awaiting for their acceptance.

Chapter 4

Discussion, conclusions, and future work

In previous sections, the discussion and conclusions for both of the proposed approaches to enhance the resolution of the PLDP problem have been already included. Nevertheless, numerous problems associated with the corresponding experiments were encountered. All these thoughts and challenges are commented in this final chapter. Last but not least, the future research avenues to keep improving the explained methods are also addressed in this chapter.

4.1 General discussion

Next, it is explained some of the most important challenges and the related decisions made along the research process to prepare not only Chapter 3 but also additional experiments that may yield relevant information for the reader's understanding of all the work done in this doctoral thesis.

As explained in Section 2.1, the accuracy of the SFs is currently limited, affecting the reliability of Docking and SBVS methods in general. In fact, there are proposal to enhance

the estimates of those SFs by replacing them with ML and DL models such as 3D CNNs, as shown in Section 2.4. Since the proposed methods in this dissertation include a force-field-based SF to provide the reward signal, the same limitations from the SFs apply to them. To overcome this difficulty, it could be used a ML-based SF although their result in terms of accuracy are normally at the same level with respect traditional SFs.

Moreover, increasing the action space was tested in both implementations in the context of the kaempferol and the cyclodextrin. In those cases, the algorithm does not converge. More specifically, the ligand tends to move around the surface of the receptor but it does not find the optimal solution in the central hole of the torus. A possible explanation for such behavior is precisely linked to the limitations of the SFs. The energy landscape that these functions generate is simply too rough for any ANN to find the global minimum energy in a reasonable amount of time. In Figure 4.1, this landscape is drawn for the kaempferol and the cyclodextrin by plotting the values of the SF from METADOCK/Bindsurf in each timestep for the center of mass of the ligand during training with movement and rotation in the three axes. Dark points are supposed to be better position in terms of Docking. Although there are quite a few black points in the center of the torus where the optimal solution is located, there are also many relative good positions around the surface of the receptor. This makes the global optima hard to be found in such steep energy landscape.

Coarse-grained energy functions [68, 145] were explored to alleviate this problem and make the energy landscape smoother, in a similar way to Figure 4.2. Unfortunately, this kind of energy functions needs to shape the Docking scene with its own coarse-grained model. We found such models too complex to be applied to the proposed methods based on RL because they required digging into more accurate approaches of Molecular Dynamics.

There is another problem related to the action space. The basic system described

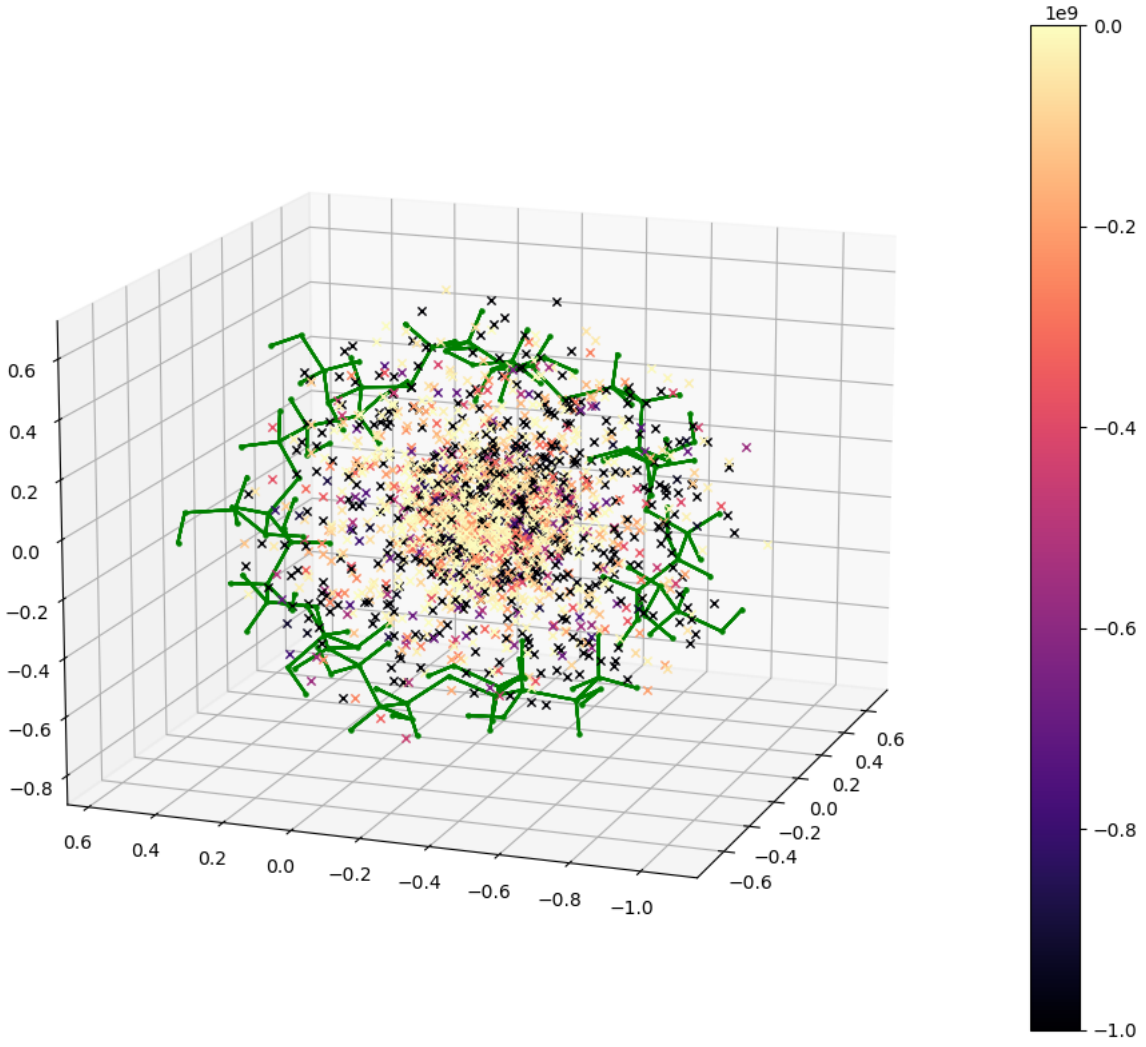

Figure 4.1: Energy landscape for the kaempferol and the cyclodextrin. Each point represents the value of the SF from METADOCCK / Bindsurf for the ligand whose center of mass is in that position at that very moment. Thus, black and darker values refer to better Docking positions (i.e. with lower energy), while points in yellow are associated with poor Docking positions (higher energy).

in Section 3.1 accepts up to 12 possible actions: translation and rotation in the three axes forwards and backwards. Additionally, it would be necessary to include molecular folding to dock the ligand more precisely and be at par with most of the state-of-the-art Docking methods. This would entail adding 3 extra degrees of freedoms or 6 actions per rotatable bond—move and rotate the molecule in the three axes forwards and backwards. The number of these rotatable bonds depends on the ligand in question and the criterion

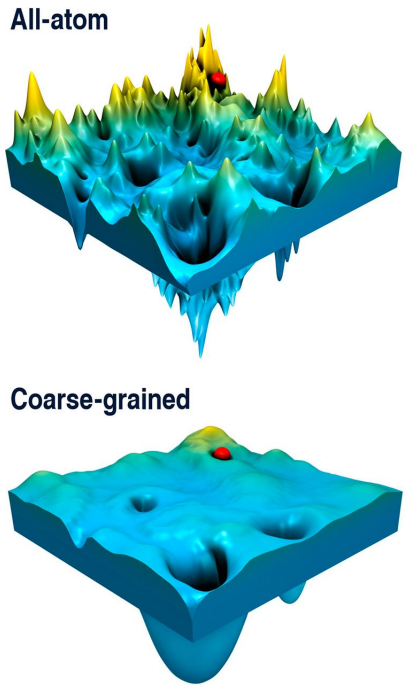

Figure 4.2: All-atom versus coarse-grained energy landscape (image and caption taken from Kmiecik et al. [68]). The figure illustrates the effect of the smoothening of the energy landscape in a coarse-grained model as compared to an all-atom model. The flattening enables efficient exploration of the energy landscape in search for the global minima, while avoiding traps in the local minima.

followed for molecular modelling. In practice, it is not normally larger than 12 although some Docking software like AutoDock Vina allow up to 40 rotatable bonds. Thus, for a given candidate with 12 rotatable bonds, for instance, the size of the action space could rise up to 84.

This entails two problems. First, the number of rotatable bonds depends on the ligand in question but the action space in RL is fix by definition. Therefore, some artifact should be applied here to overcome this hurdle. For example, the number of rotatable bonds could set to just two at the expense of accuracy. Second, a large action space is a serious problem in RL and value-based policies [85]. In the common case that the value function is a parameterized function taking both state and action as input, A evaluations are necessary to choose an action (see Section 2.2). This quickly becomes

intractable, especially if the parameterized function is costly to evaluate, as is the case with deep ANNs [30]. One key question here would be if 84 actions, for example, could be considered as too many to be handled for the DQN algorithm. To the best of our knowledge, there is no fix number of actions in the literature from which the action space could be officially considered too large. That limit seems to be context/problem dependent. Regardless, an alternative to alleviate this problem could be to consider the action space as continuous. Consequently, a policy gradient method such as A3C or DDPG could be used instead of DQN. This would halve the number of possible actions but there would be still many of them if the rotatable bonds were considered. A large number of entirely different actions, each with different results, would still require much exploration to determine the optimal combinations of action-state pairs.

The next challenges are focused on the implementation of MVDQN. Even though the ligand is not capable of finding the optimal solution for all starting positions (see Figure 3.13) in the experiment conducted in Section 3.3, a more complex setting based on seven actions and three color views was also explored. Those seven actions include movement in the three axes forwards and backwards plus an empty action. This empty or no-input action is added to allow for more optimal behavior by the ligand—if the agent finds the solution, the best action is to stay put—as in Mnih et al. [100]. Furthermore, the Multi-View CNN architecture really comes into play by increasing the number of views to three. These images are generated by PyMol in each timestep from the top, front, and lateral perspectives by zooming the ligand at its current position in the binding site. Also, the number of channels in the CNNs are increased to 3 in order to use color images. This is important since molecules are colored by atom type, being this feature relevant in Docking.

As stated above when discussing the shape of the energy landscape, MVDQN is not able to find the optimal solution in this more complex setting even in the training phase.

The ligand tends to move around the surface of the cyclodextrin, getting stuck in local optima, and it is not forced by the algorithm to explore the hole in the center to find the optimal solution. In addition to the problems mentioned above, we think that there are three other challenges in this new setting. The first one concerns sampling inefficiency. It was observed that generating images of the Docking scene in each timestep of the algorithm is too expensive for PyMol, specially with large receptors. For example, the host molecule fa10 (included in the DUD-E dataset [104]), has 3,638 atoms, quite a few more than the 146 of the beta-cyclodextrin. The visualizer spends almost 6 seconds to generate the three images with a resolution of 84 by 84 pixels. This drawback slows down the learning process in excess considering that images have to be constantly generated during training. A possible solution would be to limit the rendering of the Docking scene to the binding site. But, as explained in Section 2.3, it is not currently clear if only considering the structure of the binding site is adequate to compute binding affinity in a precise manner. Nonetheless, it is also true that the SF used to compute the reward function already includes every single atom from both the ligand and the receptor to calculate the binding energy. In other words, in the system it is indirectly considered the whole structure of the receptor through the SF.

The second challenge is also related to the size of the images and, by extension, to the size of the model. In Section 3.3.3 it is proposed to add more perspectives to the input data, increase the number of channels to include important chemical knowledge about the Docking process, or to increase the resolution of the images to provide sharper images to the neural network. Leaving aside the benefits of such measures to improve the results in the prediction phase, they also increase the size of the states and the trainable parameters. For example, in the simplified scenario used in MVDQN based on 2 actions, 1 view, and gray-scale images of 84 by 84 pixels, the total input size is 0.03 MB and the trainable parameters are 3,285,667. However, with 3 views and color images of 84 by 84

pixels, the input size is 0.24 MB and the number of parameters is 3,441,635. Likewise, for 3 views and color images of 128 by 128 pixels, the input size raises up to 0.56 MB and the the number of parameters to 9,667,555. Finally, for 8 views (top, bottom, front, back, left, and right standpoints plus two isometric views) and gray-scale images of 84 by 84 pixels, the input size is 0.22 MB and the trainable parameters are 3,788,547. In summary, it can be concluded that the image resolution leads to the greatest increase in the size of the input data and, above all, in the number of parameters to be trained. Note that, the number of views also increases the size of the input data but not as much as the image resolution. Nonetheless, as discussed earlier, increasing the number of views entails a sharp increase in the workload of the molecular visualizer. It should be pointed out as well that 0.22 MB, for instance, may not seem too much information for a modern server. But that amount of data correspond to a experience tuple to be stored in the experience replay buffer of the Deep Q-Network in each timestep. If a buffer size of 1 million is set, for example, then the total size of the buffer once filled up will be 222 GB and that amount of data is stored in the main memory. So, this calculus should be performed carefully before including additional information in the input data.

The third issue is related to molecular overlapping. Initially, this challenge was solved by using a transparency effect manually applied to the receptor in PyMol when generating the different views of the Docking scene. More specifically, this can be handle by using the clip command to adjust the slab thickness in the molecular visualizer. This can be correctly applied to small receptors such as the beta-cyclodextrin. Nonetheless, it can hardly be applied automatically in PyMol for ligand-receptor pairs with large receptors by just applying a fix parameter for the slab thickness. If that fix parameter is employed, many of the generated images turn out to be defective or non-informative with respect to the Docking state, as displayed in Figure 4.3. For example, the surface of the receptor does not show properly in many samples, hiding the ligand sometimes and completely

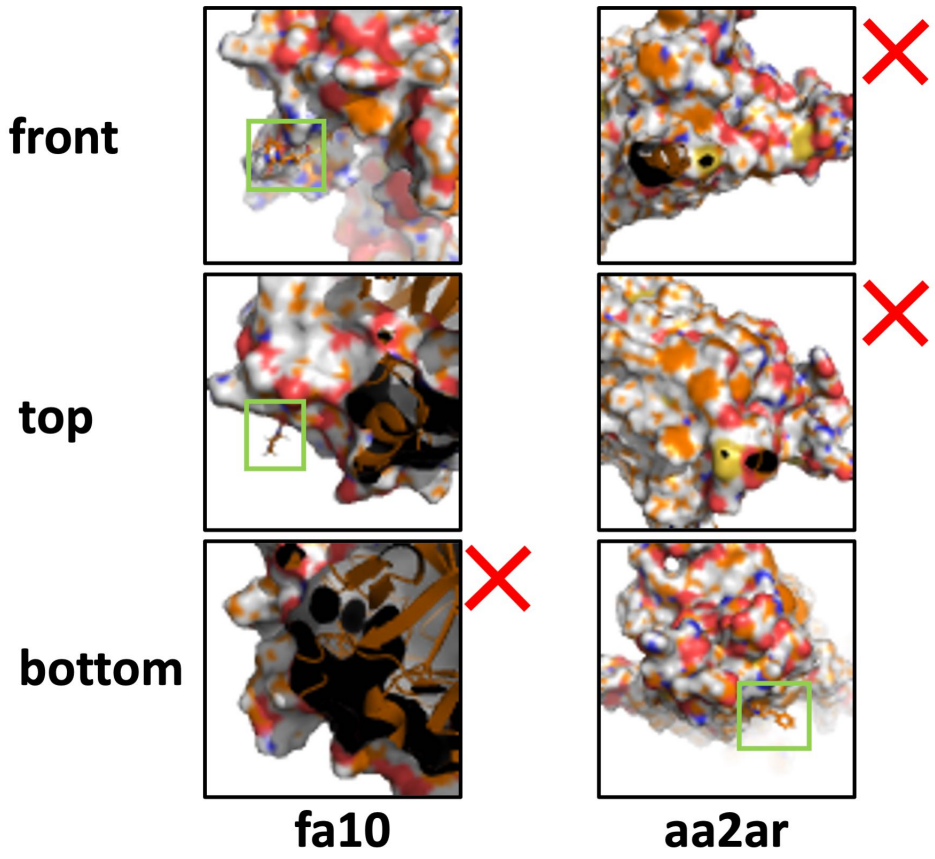

Figure 4.3: Examples of defective or non informative images of the Docking scene in large receptors included in the DUD-E banchmark (fa10 and aa2ar). Images were generated with PyMol. Those views marked with a red cross on the right side are considered as defective. If not, the ligand is marked with a green box.

vanishing in some others. Consequently, this limitation makes this approach unfeasible to be applied automatically for any given ligand-receptor pair. Maybe a supervised learning approach could be adopted, so the image would be carefully hand-picked in advance. However, such an implementation would entail a completely different method far from Deep RL and, therefore, the entire system would need to be reconsidered.

To overcome this last challenge, a new approach changing the images for molecular encoding by atomic point clouds was investigated as well. Instead of feeding pixels to the neural network, the 3D atomic coordinates plus the atom type can be included in the input data. Actually, this way of representing molecules could be regarded as a natural

extension of the feature vectors. However, point clouds were considered as intractable until recently due to their irregular, unstructured, and unordered nature. These characteristics make this kind of representations unmanageable for standard feedforward ANNs. Nevertheless, in 2017, Qi et al. [117] developed a method able to cope with this type of input data in the context of 3D classification and segmentation. Since then, a multitude of alternative models have been published [43]. It is a growing research topic indeed. In fact, there are already some works applied in drug discovery [23, 83, 93].

This new implementation based on point clouds is already built although it needs further improvements in its current state. In particular, a combination of the PointNet++ model [118] and the DQN Dueling architecture [166] was developed to select the optimal action for the agent in each timestep. The SF from METADOCK/Bindsurf is being currently vectorized to deal with larger receptor more rapidly.

4.2 Conclusions

As a final balance of the research developed in this doctoral thesis, it should remarked that the specific objectives stated in Section 1.2 have been mostly attained. Only the Specific objective 3 is not completely fulfilled. In other words, a basic system based on a Deep Reinforcement Learning algorithm (DQN) is built to solve the PLDP problem. In addition, this core system is implemented with two alternative ways for molecular encoding. The first one, QN-Docking, is based on a feature vector and is successfully tested with a relative small receptor and limited action spaces. The second implementation, MVDQN, is based on 2D drawings of the Docking scene and it is also test in the same setting than QN-Docking. The results of this last method are mixed. Apparently, the agent steadily learns to make better decisions over the training episode. However, it fails to identify the optimal solution when it starts from a different position where it was originally trained.

These mixed results with respect to MVDQN should be regarded as a golden opportunity to keep investigating in this captivating line of research involving the intersection of the Docking problem and Deep RL algorithms. The future avenues of research commented in the next section provide a rough idea of the many improvements that can be carried out. In effect, despite all these achievements, the ultimate objective (accelerating the resolution of the PLDP problem for any given ligand-receptor pair in comparison with traditional Docking methods) is far to be completed. This is something expected and understandable considering that it is a very ambitious and challenging and, therefore, it should be regarded as a long-term objective. But the effort will undoubtedly be worthwhile because, Is there a more rewarding undertaking than developing faster computational methods to sooner deliver medicines to patients?

4.3 Future work

Lastly, the next steps for expanding the work developed in this doctoral thesis are briefly enumerated. First, more actions including movement in the three axes, rotation, and molecular folding should be included for greater precision in terms of binding affinity. Second, the proposed methods need to be generalized to other ligand-host pairs beyond the kaempferol and the cyclodextrin. After solving the flaws observed in the prediction phase in MVDQN, the implementations from sections 3.2 and 3.3 could be tested with ligand-receptor pairs from datasets like PDBbind, scPDB, CSAR, DUD, and DUD-E, for instance. Third, relevant chemical information with respect to Docking should be included in the states of the algorithm beyond structural information, as explained in Section 3.3.3. For example, in the approach based on images for molecular encoding, the number of channels in the CNN could be increased to contain information about atom types, partial charges, pharmacophoric properties, atom connections, hybridization,

aromaticity, amino acid types, etc. The feature vector in QN-Docking could also be extended with such chemical knowledge.

Fourth, it has been observed that even for a Docking local optimization problem the state space is still extremely wide for a RL problem in spite of the use of ANNs. Consequently, a traditional Docking method could be used a few iterations to get the ligand closer to the optimal location. In a second step, QN-Docking could perform a fine-grained Docking to finish fitting the ligand into that optimal position. Fifth, a more systematic hyperparameter analysis could be performed by using different strategies and algorithms e.g. random search, grid search, HyperOpt, Bayes optimization, Optuna, etc. Tools like Ray Tune [84] or Microsoft's Neural Network Intelligence could be adapted to conduct the corresponding tests. Finally, as discussed in Section 4.1, the 2D drawings used in MVDQN for molecular encoding should be improved by including more perspectives of the Docking scene, increasing the number of channels, and/or the resolution of the images. But at the same time, a balance should be struck between accurate molecular representations and sample generation efficiency.

Bibliography

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[2] Hossam M Ashtawy and Nihar R Mahapatra. A comparative assessment of ranking accuracies of conventional and machine-learning-based scoring functions for protein-ligand binding affinity prediction. *IEEE/ACM Transactions on computational biology and bioinformatics*, 9(5):1301–1313, 2012.

[3] Magdalena Bacilieri and Stefano Moro. Ligand-based drug design methodologies in drug discovery process: an overview. *Current drug discovery technologies*, 3(3): 155–165, 2006.

[4] Pedro J Ballester, Adrian Schreyer, and Tom L Blundell. Does a more precise chemical description of protein–ligand complexes lead to more accurate prediction of binding affinity? *Journal of chemical information and modeling*, 54(3):944–955, 2014.

[5] Rohit Batra, Henry Chan, Ganesh Kamath, Rampi Ramprasad, Mathew J Cherukara, and Subramanian KRS Sankaranarayanan. Screening of therapeutic agents for covid-19 using machine learning and ensemble docking studies. *The journal of physical chemistry letters*, 11(17):7058–7065, 2020.

[6] Bo Ram Beck, Bonggun Shin, Yoonjung Choi, Sungsoo Park, and Keunsoo Kang. Predicting commercially available antiviral drugs that may act on the novel coronavirus (sars-cov-2) through a drug-target interaction deep learning model. *Computational and structural biotechnology journal*, 18:784–790, 2020.

[7] Andreas Bender and Isidro Cortes-Ciriano. Artificial intelligence in drug discovery: what is realistic, what are illusions? part 2: a discussion of chemical and biological data used for ai in drug discovery. *Drug Discovery Today*, 2021.

[8] Nurken Berdigaliyev and Mohamad Aljofan. An overview of drug discovery and development. *Future medicinal chemistry*, 12(10):939–947, 2020.

[9] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[10] Thomas Blaschke, Marcus Olivecrona, Ola Engkvist, Jürgen Bajorath, and Hongming Chen. Application of generative autoencoder in de novo molecular design. *Molecular informatics*, 37(1-2):1700123, 2018.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[12] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715): 547, 2018.

[13] Zixuan Cang and Guo-Wei Wei. Topologynet: Topology based deep convolutional

and multi-task neural networks for biomolecular property predictions. *PLoS computational biology*, 13(7):e1005690, 2017.

[14] Zixuan Cang and Guo-Wei Wei. Integration of element specific persistent homology and machine learning for protein-ligand binding affinity prediction. *International journal for numerical methods in biomedical engineering*, 34(2):e2914, 2018.

[15] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. The rise of deep learning in drug discovery. *Drug Discovery Today*, 23 (6):1241–1250, 2018.

[16] Lieyang Chen, Anthony Cruz, Steven Ramsey, Callum J Dickson, Jose S Duca, Viktor Hornak, David R Koes, and Tom Kurtzman. Hidden bias in the dud-e dataset leads to misleading performance of deep learning in structure-based virtual screening. *PloS one*, 14(8):e0220113, 2019.

[17] Vladimir Chupakhin, Gilles Marcou, Helena Gaspar, and Alexandre Varnek. Simple ligand–receptor interaction descriptor (silirid) for alignment-free binding site comparison. *Computational and structural biotechnology journal*, 10(16):33–37, 2014.

[18] Alexis Cook. Deep reinforcement learning nanodegree. `https://github.com/udacity/deep-reinforcement-learning`, 2018.

[19] C Da and D Kireev. Structural protein–ligand interaction fingerprints (splif) for structure-based virtual screening: method and benchmark study. *Journal of chemical information and modeling*, 54(9):2555–2561, 2014.

[20] Franck Da Silva, Jeremy Desaphy, and Didier Rognan. Ichem: a versatile toolkit for detecting, comparing, and predicting protein–ligand interactions. *ChemMedChem*, 13(6):507, 2018.

[21] Laurianne David, Josep Arús-Pous, Johan Karlsson, Ola Engkvist, Esben Jannik Bjerrum, Thierry Kogej, Jan M Kriegl, Bernd Beck, and Hongming Chen. Applications of deep-learning in exploiting large-scale and heterogeneous compound data in industrial pharmaceutical research. *Frontiers in pharmacology*, 10:1303, 2019.

[22] Laurianne David, Amol Thakkar, Rocío Mercado, and Ola Engkvist. Molecular representations in ai-driven drug discovery: a review and practical guide. *Journal of Cheminformatics*, 12(1):1–22, 2020.

[23] Ryan S DeFever, Colin Targonski, Steven W Hall, Melissa C Smith, and Sapna Sarupria. A generalized deep learning approach for local structure identification in molecular simulations. *Chemical science*, 10(32):7503–7515, 2019.

[24] Zhan Deng, Claudio Chuaqui, and Juswinder Singh. Structural interaction fingerprint (sift): a novel method for analyzing three-dimensional protein- ligand binding interactions. *Journal of medicinal chemistry*, 47(2):337–344, 2004.

[25] Jeremy Desaphy, Eric Raimbaud, Pierre Ducrot, and Didier Rognan. Encoding protein–ligand interaction patterns in fingerprints and graphs. *Journal of chemical information and modeling*, 53(3):623–637, 2013.

[26] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

[27] Tanja Dimitrov, Christoph Kreisbeck, Jill S Becker, Alán Aspuru-Guzik, and Semion K Saikin. Autonomous molecular design: then and now. *ACS applied materials & interfaces*, 11(28):24825–24836, 2019.

[28] Dong Dong, Zhijian Xu, Wu Zhong, and Shaoliang Peng. Parallelization of molec-

ular docking: a review. *Current Topics in Medicinal Chemistry*, 18(12):1015–1028, 2018.

[29] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

[30] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[31] Ozlem Erdas-Cicek, Ali Osman Atac, A Selen Gurkan-Alp, Erdem Buyukbingol, and Ferda Nur Alpaslan. Three-dimensional analysis of binding sites for predicting binding affinities in drug design. *Journal of chemical information and modeling*, 59 (11):4654–4662, 2019.

[32] Evan N Feinberg, Debnil Sur, Zhenqin Wu, Brooke E Husic, Huanghao Mai, Yang Li, Saisai Sun, Jianyi Yang, Bharath Ramsundar, and Vijay S Pande. Potentialnet for molecular property prediction. *ACS central science*, 4(11):1520–1530, 2018.

[33] Leonardo G Ferreira, Ricardo N Dos Santos, Glaucius Oliva, and Adriano D Andricopulo. Molecular docking and structure-based drug design strategies. *Molecules*, 20(7):13384–13421, 2015.

[34] Emil Fischer. Einfluss der configuration auf die wirkung der enzyme. *Berichte der Deutschen Chemischen Gesellschaft*, 27(3):2985–2993, 1894.

[35] Paul G Francoeur, Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B Iovanisci, Ian Snyder, and David R Koes. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of Chemical Information and Modeling*, 60(9):4200–4215, 2020.

[36] Garrett B Goh, Charles Siegel, Abhinav Vishnu, Nathan O Hodas, and Nathan Baker. Chemception: a deep neural network with minimal chemistry knowledge matches the performance of expert-developed qsar/qspr models. *arXiv preprint arXiv:1706.06689*, 2017.

[37] Garrett B Goh, Charles Siegel, Abhinav Vishnu, Nathan Hodas, and Nathan Baker. How much chemistry does a deep neural network need to know to make accurate predictions? In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1340–1349. IEEE, 2018.

[38] Joseph Gomes, Bharath Ramsundar, Evan N Feinberg, and Vijay S Pande. Atomic convolutional networks for predicting protein-ligand binding affinity. *arXiv preprint arXiv:1703.10603*, 2017.

[39] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[40] Adam Gonczarek, Jakub M Tomczak, Szymon Zaręba, Joanna Kaczmar, Piotr Dąbrowski, and Michał J Walczak. Interaction prediction in structure-based virtual screening using deep learning. *Computers in Biology and Medicine*, 100:253–258, 2018.

[41] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[42] Gabriel L Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro L Cunha-Farias, and Alán Aspuru-Guzik. Objective-reinforced generative ad-

versarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

[43] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[44] Anvita Gupta, Alex T Müller, Berend JH Huisman, Jens A Fuchs, Petra Schneider, and Gisbert Schneider. Generative recurrent networks for de novo drug design. *Molecular informatics*, 37(1-2):1700111, 2018.

[45] Rohan Gupta, Devesh Srivastava, Mehar Sahu, Swati Tiwari, Rashmi K Ambasta, and Pravir Kumar. Artificial intelligence to deep learning: Machine intelligence approach for drug discovery. *Molecular Diversity*, pages 1–46, 2021.

[46] Philip J Hajduk and Jonathan Greer. A decade of fragment-based drug design: strategic advances and lessons learned. *Nature Reviews Drug discovery*, 6(3):211–219, 2007.

[47] Ameya Harmalkar and Jeffrey J Gray. Advances to tackle backbone flexibility in protein docking. *Current Opinion in Structural Biology*, 67:178–186, 2021.

[48] Nafisa M Hassan, Amr A Alhossary, Yuguang Mu, and Chee-Keong Kwoh. Protein-ligand blind docking using quickvina-w with inter-process spatio-temporal integration. *Scientific reports*, 7(1):1–13, 2017.

[49] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[50] Joshua Hochuli, Alec Helbling, Tamar Skaist, Matthew Ragoza, and David R Koes. Visualizing convolutional neural network protein-ligand scoring. *Journal of Molecular Graphics and Modelling*, 84:96–108, 2018.

[51] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[52] Baldomero Imbernón, José M. Cecilia, Horacio, and Domingo Giménez. Metadock: A parallel metaheuristic schema for virtual screening methods. *The International Journal of High Performance Computing Applications*, 32(6):1–15, 2017.

[53] Baldomero Imbernón, Antonio Serrano, Andrés Bueno-Crespo, José L Abellán, Horacio Pérez-Sánchez, and José M Cecilia. Metadock 2: A high-throughput parallel metaheuristic scheme for molecular docking. *Bioinformatics*, 2018.

[54] Baldomero Imbernón, Antonio Serrano, Andrés Bueno-Crespo, José L Abellán, Horacio Pérez-Sánchez, and José M Cecilia. Metadock 2: a high-throughput parallel metaheuristic scheme for molecular docking. *Bioinformatics*, 37(11):1515–1520, 2021.

[55] John J Irwin and Brian K Shoichet. ZINC–a free database of commercially available compounds for virtual screening. *Journal of Chemical Information and Modeling*, 45 (1):177–182, 2005.

[56] Mohammad Behdad Jamshidi, Ali Lalbakhsh, Jakub Talla, Zdeněk Peroutka, Sobhan Roshani, Vaclav Matousek, Saeed Roshani, Mirhamed Mirmozafari, Zahra Malek, Luigi La Spada, et al. Deep learning techniques and covid-19 drug discovery: Fundamentals, state-of-the-art and future directions. *Emerging Technologies During the Era of COVID-19 Pandemic*, 348:9, 2021.

[57] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José M Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1645–1654. JMLR. org, 2017.

[58] Julia B Jasper, Lina Humbeck, Tobias Brinkjost, and Oliver Koch. A novel interaction fingerprint derived from per atom score contributions: exhaustive evaluation of interaction fingerprint performance in docking based virtual screening. *Journal of cheminformatics*, 10(1):1–13, 2018.

[59] Woosung Jeon and Dongsup Kim. Autonomous molecule generation using reinforcement learning and docking to develop potential novel inhibitors. *Scientific reports*, 10(1):1–11, 2020.

[60] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[61] José Jiménez, Stefan Doerr, Gerard Martínez-Rosell, Alexander S. Rose, and Gianni De Fabritiis. Deepsite: protein-binding site predictor using 3d-convolutional neural networks. *Bioinformatics*, 33(19):3036–3042, 2017.

[62] José Jiménez, Miha Skalic, Gerard Martinez-Rosell, and Gianni De Fabritiis. K deep: Protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks. *Journal of Chemical Information and Modeling*, 58(2):287–296, 2018.

[63] William L Jorgensen. The Many Roles of Computation in Drug Discovery. *Science*, 303:1813–1818, 2004. doi: 10.1126/science.1096361.

[64] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna

Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, pages 1–11, 2021.

[65] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018.

[66] Talia B Kimber, Yonghui Chen, and Andrea Volkamer. Deep learning in virtual screening: Recent applications and developments. *International Journal of Molecular Sciences*, 22(9):4435, 2021.

[67] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[68] Sebastian Kmiecik, Dominik Gront, Michal Kolinski, Lukasz Wieteska, Aleksandra Elzbieta Dawid, and Andrzej Kolinski. Coarse-grained protein models and their applications. *Chemical reviews*, 116(14):7898–7936, 2016.

[69] David R Koes, Matthew P Baumgartner, and Carlos J Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of Chemical Information and Modeling*, 53(8):1893–1904, 2013.

[70] Daniel E Koshland. Correlation of structure and function in enzyme action. *Science*, 142(3599):1533–1541, 1963.

[71] Marina Krakovsky. Reinforcement renaissance. *Commun. ACM*, 59(8):12–14, 2016.

[72] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[73] Irwin D Kuntz, Jeffrey M Blaney, Stuart J Oatley, Robert Langridge, and Thomas E Ferrin. A geometric approach to macromolecule-ligand interactions. *Journal of molecular biology*, 161(2):269–288, 1982.

[74] Samuel Lalmuanawma, Jamal Hussain, and Lalrinfela Chhakchhuak. Applications of machine learning and artificial intelligence for covid-19 (sars-cov-2) pandemic: A review. *Chaos, Solitons & Fractals*, 139:110059, 2020.

[75] Antonio Lavecchia and Carmen Di Giovanni. Virtual screening strategies in drug discovery: a critical review. *Current Medicinal Chemistry*, 20(23):2839–2860, 2013.

[76] Yann LeCun, Yoshua Bengio, and Geoffrey E Hinton. Deep learning. *Nature*, 521 (7553):436, 2015.

[77] Biao Leng, Shuang Guo, Xiangyang Zhang, and Zhang Xiong. 3d object retrieval with stacked local convolutional autoencoder. *Signal Processing*, 112:119–128, 2015.

[78] Hongjian Li, Kam-Heung Sze, Gang Lu, and Pedro J Ballester. Machine-learning scoring functions for structure-based virtual screening. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 11(1):e1478, 2021.

[79] Yan Li, Li Han, Zhihai Liu, and Renxiao Wang. Comparative assessment of scoring functions on an updated benchmark: 2. evaluation methods and general results. *Journal of Chemical Information and Modeling*, 54(6):1717–1736, 2014.

[80] Yanjun Li, Mohammad A Rezaei, Chenglong Li, and Xiaolin Li. Deepatom: a framework for protein-ligand binding affinity prediction. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 303–310. IEEE, 2019.

[81] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[82] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[83] Zhen Li, Xu Yan, Qing Wei, Xin Gao, Sheng Wang, and Shuguang Cui. Pointsite: a point cloud segmentation tool for identification of protein ligand binding atoms. *bioRxiv*, 2019. doi: 10.1101/831131. URL `https://www.biorxiv.org/content/early/2019/11/05/831131`.

[84] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[85] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[86] Jaechang Lim, Seongok Ryu, Kyubyong Park, Yo Joong Choe, Jiyeon Ham, and Woo Youn Kim. Predicting drug–target interaction using a novel graph neural network with 3d structure-embedded graph representation. *Journal of chemical information and modeling*, 59(9):3981–3988, 2019.

[87] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1992.

[88] Jie Liu and Renxiao Wang. Classification of current scoring functions. *Journal of chemical information and modeling*, 55(3):475–482, 2015.

[89] Xuhan Liu, Kai Ye, Herman WT Van Vlijmen, Adriaan P IJzerman, and Gerard JP Van Westen. An exploration strategy improves the diversity of de novo ligands

using deep reinforcement learning: a case for the adenosine a 2a receptor. *Journal of cheminformatics*, 11(1):1–16, 2019.

[90] Jiankun Lyu, Sheng Wang, Trent E Balius, Isha Singh, Anat Levit, Yurii S Moroz, Matthew J O'Meara, Tao Che, Enkhjargal Algaa, Kateryna Tolmachova, et al. Ultra-large library docking for discovering new chemotypes. *Nature*, 566(7743):224–229, 2019.

[91] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of Chemical Information and Modeling*, 55(2):263–274, 2015.

[92] Eduardo Habib Bechelane Maia, Letícia Cristina Assis, Tiago Alves de Oliveira, Alisson Marques da Silva, and Alex Gutterres Taranto. Structure-based virtual screening: From classical to artificial intelligence. *Frontiers in chemistry*, 8, 2020.

[93] Vincent Mallet. *Leveraging binding-site structure for drug discovery with point-cloud methods*. McGill University (Canada), 2019.

[94] Yasunari Matsuzaka and Yoshihiro Uesawa. A molecular image-based novel quantitative structure-activity relationship approach, deepsnap-deep learning and machine learning. *Current Issues in Molecular Biology*, 42(1):455–472, 2021.

[95] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

[96] Andrew T McNutt, Paul Francoeur, Rishal Aggarwal, Tomohide Masuda, Rocco Meli, Matthew Ragoza, Jocelyn Sunseri, and David Ryan Koes. Gnina 1.0: molecular docking with deep learning. *Journal of cheminformatics*, 13(1):1–20, 2021.

[97] Xuan-Yu Meng, Hong-Xing Zhang, Mihaly Mezei, and Meng Cui. Molecular docking: a powerful approach for structure-based drug discovery. *Current computer-aided drug design*, 7(2), 2011. ISSN 1875-6697. doi: 10.2174/157340911795677602.

[98] Kenneth M Merz Jr, Dagmar Ringe, and Charles H Reynolds. *Drug design: structure-and ligand-based approaches*. Cambridge University Press, 2010.

[99] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[100] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[101] Joseph A Morrone, Jeffrey K Weber, Tien Huynh, Heng Luo, and Wendy D Cornell. Combining docking pose rank and structure with deep learning improves protein–ligand binding mode prediction over a baseline docking approach. *Journal of chemical information and modeling*, 60(9):4170–4179, 2020.

[102] Varnavas D Mouchlis, Antreas Afantitis, Angela Serra, Michele Fratello, Anastasios G Papadiamantis, Vassilis Aidinis, Iseult Lynch, Dario Greco, and Georgia Melagraki. Advances in de novo drug design: from conventional to machine learning methods. *International journal of molecular sciences*, 22(4):1676, 2021.

[103] John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, Torsten Schwede, and Maya

Topf. *Critical Assessment of Techniques for Protein Structure Prediction (CASP), Fourteenth Round: Abstract Book*. -, 2020.

[104] Michael M Mysinger, Michael Carchia, John J Irwin, and Brian K Shoichet. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, 55(14):6582–6594, 2012.

[105] Duc Duy Nguyen, Zixuan Cang, Kedi Wu, Menglun Wang, Yin Cao, and Guo-Wei Wei. Mathematical deep learning for pose and binding affinity prediction and ranking in d3r grand challenges. *Journal of computer-aided molecular design*, 33(1): 71–82, 2019.

[106] Duc Duy Nguyen, Zixuan Cang, and Guo-Wei Wei. A review of mathematical representations of biomolecular data. *Physical Chemistry Chemical Physics*, 22(8): 4343–4367, 2020.

[107] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9(1):48, 2017.

[108] Noel M O'Boyle. Towards a universal smiles representation-a standard method to generate canonical smiles based on the inchi. *Journal of cheminformatics*, 4(1):1–14, 2012.

[109] Nataraj S Pagadala, Khajamohiddin Syed, and Jack Tuszynski. Software for molecular docking: a review. *Biophysical reviews*, 9(2):91–102, 2017.

[110] Conor D Parks, Zied Gaieb, Michael Chiu, Huanwang Yang, Chenghua Shao, W Patrick Walters, Johanna M Jansen, Georgia McGaughey, Richard A Lewis, Scott D Bembenek, et al. D3r grand challenge 4: blind prediction of protein–ligand

poses, affinity rankings, and relative binding free energies. *Journal of computer-aided molecular design*, 34(2):99–119, 2020.

[111] Janaina C Pereira, Ernesto R Caffarena, and Cicero N dos Santos. Boosting docking-based virtual screening with deep learning. *Journal of Chemical Information and Modeling*, 56(12):2495–2506, 2016.

[112] Violeta I Pérez-Nueno, Obdulia Rabal, José I Borrell, and Jordi Teixidó. Apif: a new interaction fingerprint based on atom pairs and its application to virtual screening. *Journal of chemical information and modeling*, 49(5):1245–1260, 2009.

[113] Javier Pérez-Sianes, Horacio Pérez-Sánchez, and Fernando Díaz. Virtual screening meets deep learning. *Current Computer-Aided Drug Design*, 15(1):6–28, 2019.

[114] Luca Pinzi and Giulio Rastelli. Molecular docking: Shifting paradigms in drug discovery. *International journal of molecular sciences*, 20(18):4331, 2019.

[115] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science Advances*, 4(7):eaap7885, 2018.

[116] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[117] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[118] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical

feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[119] Tong Qin, Zihao Zhu, Xiang Simon Wang, Jie Xia, and Song Wu. Computational representations of protein-ligand interfaces for structure-based virtual screening. *Expert Opinion on Drug Discovery*, 0(0):1–18, 2021.

[120] Muhammad Radifar, Nunung Yuniarti, and Enade Perdana Istyastono. Pyplif: Python-based protein-ligand interaction fingerprinting. *Bioinformation*, 9(6):325, 2013.

[121] Matthew Ragoza, Joshua Hochuli, Elisa Idrobo, Jocelyn Sunseri, and David R Koes. Protein–ligand scoring with convolutional neural networks. *Journal of Chemical Information and Modeling*, 57(4):942–957, 2017.

[122] Ahmet Sureyya Rifaioglu, Heval Atas, Maria Jesus Martin, Rengul Cetin-Atalay, Volkan Atalay, and Tunca Dogan. Recent applications of deep learning and machine intelligence on in silico drug discovery: methods, tools and databases. *Briefings in Bioinformatics*, 10, 2018.

[123] Ahmet Sureyya Rifaioglu, Esra Nalbat, Volkan Atalay, Maria Jesus Martin, Rengul Cetin-Atalay, and Tunca Doğan. Deepscreen: high performance drug–target interaction prediction with convolutional neural networks using 2-d structural compound representations. *Chemical science*, 11(9):2531–2557, 2020.

[124] Filipe Marinho Rocha, Vítor Santos Costa, and Luís Paulo Reis. From reinforcement learning towards artificial general intelligence. In *World Conference on Information Systems and Technologies*, pages 401–413. Springer, 2020.

[125] Judith M Rollinger, Hermann Stuppner, and Thierry Langer. Virtual screening for

the discovery of bioactive natural products. In *Natural Compounds as Drugs Volume I*, pages 211–249. Springer, 2008.

[126] Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall Upper Saddle River, NJ, USA:, 2002.

[127] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L Guimaraes, and Alán Aspuru-Guzik. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). *Harvard University, Chem Rxiv*, 2017. doi: 10.26434/chemrxiv.5309668.v3.

[128] I Sánchez-Linares, H Pérez-Sánchez, JM Cecilia, and JM García. Bindsurf: a fast blind virtual screening methodology on gpus. *Network Tools and Applications in Biology (NETTAB 2011), Clinical Bioinformatics*, pages 95–97, 2011.

[129] Tomohiro Sato, Teruki Honma, and Shigeyuki Yokoyama. Combining machine learning and pharmacophore-based interaction fingerprint for in silico screening. *Journal of chemical information and modeling*, 50(1):170–185, 2010.

[130] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[131] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[132] Nadine Schneider, Roger A Sayle, and Gregory A Landrum. Get your atoms in order–an open-source implementation of a novel and robust molecular canonicalization algorithm. *Journal of chemical information and modeling*, 55(10):2111–2120, 2015.

[133] John S Schreck, Connor W Coley, and Kyle JM Bishop. Learning retrosynthetic planning through simulated experience. *ACS central science*, 5(6):970–981, 2019.

[134] Schrödinger, LLC. The PyMOL molecular graphics system, version 2.4. -, November 2015.

[135] Saptarshi Sengupta, Sanchita Basak, Pallabi Saikia, Sayak Paul, Vasilios Tsalavoutis, Frederick Atiah, Vadlamani Ravi, and Alan Peters. A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowledge-Based Systems*, 194:105596, 2020.

[136] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[137] Antonio Serrano, Baldomero Imbernón, Horacio Pérez-Sánchez, José M Cecilia, Andrés Bueno-Crespo, and José L Abellán. Accelerating drugs discovery with deep reinforcement learning: An early approach. In *Proceedings of the 47th International Conference on Parallel Processing Companion*, page 6. ACM, 2018.

[138] Antonio Serrano, Baldomero Imbernón, Horacio Pérez-Sánchez, José M Cecilia, Andrés Bueno-Crespo, and José L Abellán. Qn-docking: An innovative molecular docking methodology based on q-networks. *Applied Soft Computing*, 96:106678, 2020.

[139] Brian K Shoichet, Irwin D Kuntz, and Dale L Bodian. Molecular docking using shape descriptors. *Journal of Computational Chemistry*, 13(3):380–397, 1992.

[140] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George

Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[141] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[142] Austin Silveria. Unity banana collection. `https://github.com/austinsilveria/Banana-Collection-DQN`, 2018.

[143] Miha Skalic, Alejandro Varela-Rial, José Jiménez, Gerard Martínez-Rosell, and Gianni De Fabritiis. Ligvoxel: inpainting binding pockets using 3d-convolutional neural networks. *Bioinformatics*, 35(2):243–250, 2018.

[144] Miha Skalic, Gerard Martínez-Rosell, José Jiménez, and Gianni De Fabritiis. Playmolecule bindscope: large scale cnn-based virtual screening on the web. *Bioinformatics*, 35(7):1237–1238, 2019.

[145] Paulo CT Souza, Riccardo Alessandri, Jonathan Barnoud, Sebastian Thallmair, Ignacio Faustino, Fabian Grünewald, Ilias Patmanidis, Haleh Abdizadeh, Bart MH Bruininks, Tsjerk A Wassenaar, et al. Martini 3: a general purpose force field for coarse-grained molecular dynamics. *Nature methods*, 18(4):382–388, 2021.

[146] Niclas Ståhl, Goran Falkman, Alexander Karlsson, Gunnar Mathiason, and Jonas

Bostrom. Deep reinforcement learning for multiparameter optimization in de novo drug design. *Journal of chemical information and modeling*, 59(7):3166–3176, 2019.

[147] Marta M Stepniewska-Dziubinska, Piotr Zielenkiewicz, and Pawel Siedlecki. Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. *Bioinformatics*, 34(21):3666–3674, 2018.

[148] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[149] Jocelyn Sunseri, Jonathan E King, Paul G Francoeur, and David R Koes. Convolutional neural network scoring and minimization in the d3r 2017 community challenge. *Journal of Computer-Aided Molecular Design*, 33(1):19–34, 2019.

[150] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044, 1996.

[151] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[152] Dominique Sydow, Lindsey Burggraaff, Angelika Szengel, Herman WT van Vlijmen, Adriaan P IJzerman, Gerard JP van Westen, and Andrea Volkamer. Advances and challenges in computational target prediction. *Journal of chemical information and modeling*, 59(5):1728–1742, 2019.

[153] Roberto Todeschini and Viviana Consonni. *Handbook of molecular descriptors*, volume 11. John Wiley & Sons, 2008.

[154] Anh-Tien Ton, Francesco Gentile, Michael Hsing, Fuqiang Ban, and Artem Cherkasov. Rapid identification of potential inhibitors of sars-cov-2 main protease by deep docking of 1.3 billion compounds. *Molecular informatics*, 39(8):2000028, 2020.

[155] Pedro HM Torres, Ana CR Sodero, Paula Jofily, and Floriano P Silva-Jr. Key topics in molecular docking for drug design. *International journal of molecular sciences*, 20 (18):4574, 2019.

[156] Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31(2):455–461, 2010.

[157] Masashi Tsubaki, Kentaro Tomii, and Jun Sese. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 35(2):309–318, 2019.

[158] Kathryn Tunyasuvunakool, Jonas Adler, Zachary Wu, Tim Green, Michal Zielinski, Augustin Žídek, Alex Bridgland, Andrew Cowie, Clemens Meyer, Agata Laydon, Sameer Velankar, Gerard J. Kleywegt, Alex Bateman, Richard Evans, Alexander Pritzel, Michael Figurnov, Olaf Ronneberger, Russ Bates, Simon A. A. Kohl, Anna Potapenko, Andrew J. Ballard, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Ellen Clancy, David Reiman, Stig Petersen, Andrew W. Senior, Koray Kavukcuoglu, Ewan Birney, Pushmeet Kohli, John Jumper, and Demis Hassabis. Highly accurate protein structure prediction for the human proteome. *Nature*, 2021. doi: 10.1038/s41586-021-03828-1. URL `https://doi.org/10.1038/s41586-021-03828-1`.

[159] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[160] Javier Vázquez, Manel López, Enric Gibert, Enric Herrero, and F Javier Luque. Merging ligand-based and structure-based methods in drug discovery: An overview of combined virtual screening approaches. *Molecules*, 25(20):4723, 2020.

[161] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[162] Izhar Wallach, Michael Dzamba, and Abraham Heifets. Atomnet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*, 2015.

[163] Hao-nan Wang, Ning Liu, Yi-yun Zhang, Da-wei Feng, Feng Huang, Dong-sheng Li, and Yi-ming Zhang. Deep reinforcement learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, pages 1–19, 2020.

[164] Renxiao Wang, Luhua Lai, and Shaomeng Wang. Further development and validation of empirical scoring functions for structure-based binding affinity prediction. *Journal of computer-aided molecular design*, 16(1):11–26, 2002.

[165] Yan-Bin Wang, Zhu-Hong You, Shan Yang, Hai-Cheng Yi, Zhan-Heng Chen, and Kai Zheng. A deep learning-based method for drug-target interaction prediction based on long short-term memory neural network. *BMC medical informatics and decision making*, 20(2):1–9, 2020.

[166] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando

Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[167] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning. In Maria F Balcan and Kilian Q Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `http://proceedings.mlr.press/v48/wangf16.html`.

[168] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4): 279–292, 1992.

[169] Julia Weber, Janosch Achenbach, Daniel Moser, and Ewgenij Proschak. Vammpire: a matched molecular pairs database for structure-based drug design and optimization. *Journal of medicinal chemistry*, 56(12):5203–5207, 2013.

[170] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[171] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[172] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[173] Yinqiu Xu, Hequan Yao, and Kejiang Lin. An overview of neural networks for drug discovery and the inputs used. *Expert Opinion on Drug Discovery*, 13(12):1091–1102, 2018.

[174] Hongming Zhang and Tianyang Yu. Taxonomy of reinforcement learning algorithms. In *Deep Reinforcement Learning*, pages 125–133. Springer, 2020.

[175] Zizhao Zhang, Haojie Lin, Xibin Zhao, Rongrong Ji, and Yue Gao. Inductive multi-hypergraph learning and its application on view-based 3d object classification. *IEEE Transactions on Image Processing*, 27(12):5957–5968, 2018.

[176] Fangqiang Zhu, Xiaohua Zhang, Jonathan E Allen, Derek Jones, and Felice C Lightstone. Binding affinity prediction by pairwise function based on neural network. *Journal of chemical information and modeling*, 60(6):2766–2772, 2020.