Serrano, A.; Imbernón, B.; Pérez-Sánchez, H.; Cecilia-Canales, JM.; Bueno-Crespo, A.; Abellán, JL. (2020). QN-Docking: An innovative molecular docking methodology based on Q-Networks. Applied Soft Computing. 96:1-12. https://doi.org/10.1016/j.asoc.2020.106678

DQN-Docking: An innovative molecular docking methodology based on Deep Q-Networks

Antonio Serrano[1,*], Baldomero Imbernón[1], Horacio Pérez-Sánchez[1], José M. Cecilia[1], Andrés Bueno-Crespo[1], José L. Abellán[1]

[1] *Bioinformatics and High Performance Computing Research Group (BIO-HPC), Computer Science Department, Universidad Católica de Murcia (UCAM), Spain*

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 are often used to predict pharmacological candidates during the first stages. One of the most widely used methods in virtual screening is molecular docking. The goal of this method is to predict the 3D conformations where a potential pharmacological candidate (also known as the ligand) binds to a given spot. Traditional docking methods are based on optimization procedures of scoring functions–i.e. mathematical functions that model molecular interactions–characterized by being computationally expensive. Thus, an alternative approach called *DQN-Docking* based on deep reinforcement learning is proposed to provide a novel framework for developing docking simulations efficiently. In particular, the developed approach is built upon a Deep Q-Network to train the ligand (the agent) to find its optimal interaction. The experimental section is centered on the cyclodextrin as the host molecule, and it is shown that once the agent is trained, it is able to find the optimal solution independently of its starting position. These results suppose a valuable milestone in developing a faster and effective method for

*Corresponding author

Email address: aserrano7@ucam.edu (Antonio Serrano)

The authors declare that there is no conflict of interests regarding the publication of this article.

docking simulations.

Keywords: Deep reinforcement learning, Deep Q-Network, Structure-based drug design, Protein-ligand interactions, Molecular docking

1. Introduction

Drug development process is known for being excessively expensive, long, and difficult. In fact, it often takes an average of 10-15 years from the initial steps—where thousands of pharmacological candidates are considered—to regulator's approval [10]. Drug discovery, in particular, encompasses the first stages of the entire drug development pipeline previous to chemical synthesis of a selected group of pharmacological candidates. In the last decade, Virtual Screening (VS) methods have been heavily used to speed up this part of drug development. They consist of molecular simulations performed in a fast, precise fashion to recreate the atomic interactions among molecules. One of the most effective methods in VS is docking [20], applied to solve the so-called Protein-ligand Docking Prediction (PLDP) problem. Such a method is based on the exploration of ligand conformations adopted within the binding sites of macromolecular targets [3]. As a result, docking not only shortens the research-to-market cycles but also leads to enormous cost savings [18]. The other side of the coin, however, is its intensive computational expense, which demands powerful high-performance computing platforms programmed by means of advanced parallel programming models [13].

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). DL is a family of ML algorithms based on Artificial Neural Networks (ANNs) with many sequential layers of simple computing units (artificial neurons) for learning data representations [43]. The adjective "deep" refers to the fact that the given ANN is made of many layers, in contrast with shallow ANNs. 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 applied to a wide range of contexts such as speech recognition, machine translation, image recognition, and self-driving cars [26], to name just a few.

As a consequence in the last lustrum, there has been an increasing and vivid research trend of DL applied to drug discovery [2; 39; 5; 6], specially since the astounding results achieved in Merck Kaggle and NIH Tox21 data challenges. Thus, [2] identifies five major topics in this regard: molecular property and activity prediction; molecular de novo design; reactions prediction, planning synthesis, and retrosynthetic analysis; protein-ligand interactions prediction; and biological imaging analysis. To the extent of our knowledge, the first two topics along with biological imaging analysis are by far the most fruitful within drug discovery in terms of achieved performance and the number of publications.

With respect to the protein-ligand interactions prediction, which involves the resolution of the PLDP problem, there have been also some remarkable advancements [57; 37; 16]. In most of them, 3D and 2D grids generated from molecular coordinates are used as inputs for Convolutional Neural Networks (CNNs) [23], a particular ANN architecture that performs specially well in image recognition tasks. CNNs are then used as Scoring Functions (SFs) in active/inactive detection in protein-ligand complexes, or in score pose and binding affinity prediction, for example.

In addition to DL, Reinforcement Learning (RL) is another renowned subdiscipline in ML. It pursues the goal of training an agent to interact with a given environment to maximize some notion of cumulative reward in the long term [53]. 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 is currently living a renaissance thanks to more powerful computers, new algorithmic techniques, mature software packages and architectures, and strong financial support [28]. 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 (DRL). These algorithms integrate an ANN

in some of the basic components of an RL system such as the policy function, the value function, or the transition model. Although there had been several important attempts to integrate DL and RL algorithms in a single system before, nobody had been successful until the advent of the striking breakthroughs of Deep Q-network (DQN) [32] and AlphaGo [47; 48]. Consequently, DRL is expected to revolutionize the field of Artificial Intelligence in the next years [1]. Moreover, RL and DRL have been applied in drug discovery as well, in particular in molecular de novo design [15; 33; 36], retrosynthesis [44], and inverse-design chemistry [41].

As explained by [2], the results so far in drug discovery applying DL and RL methods show better performance for certain tasks like bioactivity prediction through multitask learning, de novo molecular design, and image analysis. In respect of molecular docking, though, the same authors pointed out that despite of the hopeful results obtained with CNNs, it is not clear whether they will become a real, much better alternative to traditional methods. Nevertheless, we do believe that DRL can definitely enhance and accelerate the resolution of the PLDP problem by harnessing the power of ANNs as function approximators to train the agent to navigate and find the optimal solution. With this aim, in this paper we propose for the first time an approach based on a DRL scheme. This work is an extension of our previous effort [45]. To the best of our knowledge, there is no other previous study on protein-ligand interactions prediction based on DRL apart from our aforementioned work [24; 62]. In particular, we introduce a system that involves the application of the DQN algorithm to train the ligand to search for the optimal interaction site in a docking scenario guided by a force-field-based SF. Thus, the main contribution of this work could be considered the creation of a novel docking method compared to current state-of-the-art alternatives. We are convinced that this alternative approach based on DRL could overcome some of the weaknesses of those more traditional methods, which are currently unfeasible to process large chemical databases in a reasonable amount of time. As a first step to fulfill this ambitious task, we design a conceptual study with the aim of demonstrating that, once the agent has

been trained, it is able to find the optimal spot in a subsequent prediction phase regardless of its initial location. We believe that this is a promising, stimulating avenue of research that can contribute to accelerate drug discovery and sooner deliver medicines to patients urgently in need.

2. Background

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 [40]. 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 1). To do so, docking consists of two interrelated steps: (i) sampling conformations of the ligand in the binding site of the host; and (ii) accurate prediction of the interaction energy associated with those conformations using a 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 problem is called blind docking [54].

Moreover, there exist two different approaches in drug design, Structure-Based Drug Design (SBDD) and Ligand-Based Drug Design (LBDD) / Similarity-Based Drug Discovery [31; 3]. 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).

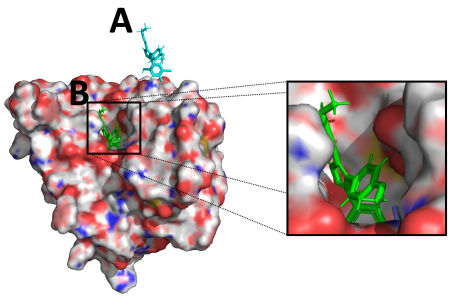

Figure 1: Illustration of the PLDP problem. The objective for the simulated ligand (A, with a blue skeleton) is to find the optimal interaction site (B, with a green skeleton) in the host surface. The optimal interaction site is zoomed for clarity sake.

Such a method broadly encompasses docking, molecular dynamics, and quantum mechanics. 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 [25]. Actually, these libraries may contain millions of compounds [14]. 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 [4], 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 [22], 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. Molecular dynamics would be the ideal way of addressing this issue. Unfortunately, this method entails a much higher computational cost, which prevents this alternative 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 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 a NP-complete problem [46] where 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. There are three types of SFs depending on the nature of the information included in their estimates: force-field-based, empirical, and knowledge-based [30]. Each of those SFs have their own benefits and drawbacks. 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 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.

2.2. Deep Q-Network

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, 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, this work is focused on non-stationary and Partially Observable Markov Decision Process (POMDP)–where the agent only is able to see part of the world state instead of the internal state (memory) to act optimally–problems.

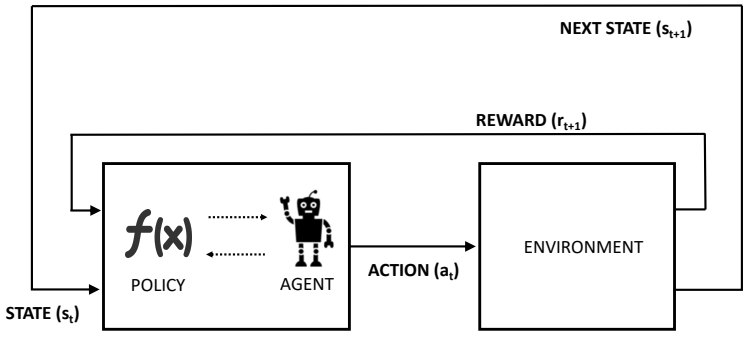

Figure 2: Representation of a typical problem in Reinforcement Learning.

In this work a DQN is applied, which is based on another well-know algorithm called Q-learning [59]. This 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 of each state when executing a certain action, instead of directly learn the optimal policy as in policy-based algorithms. More specifically, those utility values are continuously updated according to the rule in Equation 1:

$$Q'(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{1}$$

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 [52]. In particular, DQN follows an ϵ-greedy strategy as the behavior policy in order to manage the exploration/exploitation trade-off.

DQN is also an approximate solution method, not tabular. In this case, an ANN is used to approximate the Q-value function $Q(s, a|\theta)$, where θ represent 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:

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

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 [19] is chosen as the update rule for this work. In supervised learning, the loss is the difference between the actual and the predicted values. Similarly, in this context the loss is 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 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 [29], 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 experiences 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. Thus, double deep q-learning [56] 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 [58] 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 [42] 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.

3. Related work

Next, it is succinctly reviewed previous works in DL applied to docking. Note that these articles are based on supervised learning instead of RL. But as far as we are aware, these are the nearest related works to the intersection of

the PLDP problem and DRL algorithms, as intended in the current work.

As previously stated, there is a research trend in which scientists are already investigating the application of DL algorithms to protein-ligand interactions prediction [35]. Basically, they aim to substitute traditional docking SFs and ML SFs by CNNs to predict binding affinity [11; 51; 50; 7; 17], pose score [38], active/inactive molecules [57; 37; 8; 34], binding sites [16], or properties of potential ligands interacting with proteins [49].

In these models, 3D grids are the most widely used format to describe the molecules. This kind of input does not only take into account the atom coordinates but also extra information such as the atom types, partial charges, pharmacophoric and SMART properties, voxel occupancy, atom connections, hybridization, amino acid types, etc. According to this kind of input, 3D CNNs is the preferred ANN's architecture. It is worth noting that only [17] make use of deeper ANNs—in particular a variant of SqueezeNet [12]—, 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 ML methods and traditional scoring-function-based methods. For example, [57] achieve an AUC greater than 0.9 in the DUD-E benchmark, surpassing previous docking methods included in Smina [21]. But these results are constrained to 57.8% of the targets included in the benchmark. Furthermore, the CNN in [37] outperforms Autodock Vina [55] empirical SF, and ML-based SFs like RF-Score and NNScore in VS and intertarget evaluations of pose prediction. However, it performs worse at intratarget pose ranking, which is more relevant to docking. [51] 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 [50] outperforms the SFs tested in [27] 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. [17] 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.

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 is not only related to the algorithms themselves but also with the lack of gold-standard datasets in ML applied to drug discovery [39]. This lack is partly due to the heterogeneity of the information included in the drug discovery datasets. There have been some recent, laudable efforts in drug discovery to create something similar to ImageNet database in image recognition [60]. In the meantime, though, datasets like PDBbind, scPDB, CSAR, DUD, and DUD-E are some of the most widely used benchmarks in protein-ligand interactions prediction. Although some of these results obtained with CNNs are encouraging, this issue casts doubts on the ability of these DL models to consistently improve results compared to traditional SFs and other ML-based methods [2].

4. Implementation

The main contribution of the current work is to introduce a new promising method based on DRL for solving the PLDP problem. In the following lines, it is thoroughly explained how this method has been implemented. This brings about important decisions on the building blocks of DRL: defining the states, actions, states, reward function, architecture of the ANN, stop conditions, etc.

As illustrated in Figure 3, these are the major components of RL (see Section 2) associated with the PLDP problem:

- The agent, which is incarnated by the ligand.
- The value function, which indirectly determines the policy of the agent. It is represented by a standard feedforward ANN estimating the Q values.
- The actions the agent may take. There are two possible discrete actions: moving forward or backward along one spatial axis.
- The environment ε represented by the docking program METADOCK [13].
- The states and next states embodied by the mass center of the ligand plus the rotational quaternions and their norm.
- The reward based on a transformed score obtained from METADOCK SF. This reward gradually adds the SF terms according to several conditions, as depicted below.

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 reduce the search space of Q-values as much as possible to facilitate the convergence of the algorithm. In future publications, it is intended to include movement and rotation in the three axes and ligand folding. But this is far beyond a reasonable scope for the current work. Then, the selected action is passed to METADOCK, which computes the new position or state of the ligand and its corresponding score.

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 for later extensions of *DQN-Docking*. More complete representations of molecules like 3D structures [61] 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

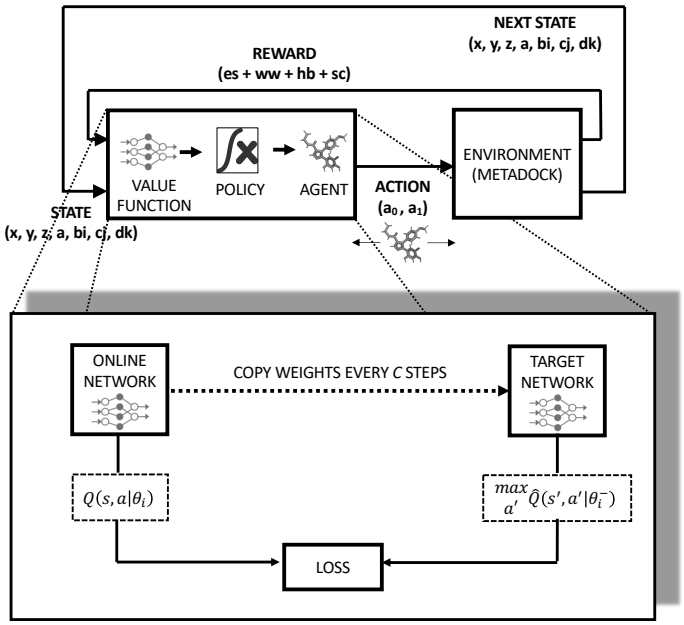

Figure 3: Operational schema of *DQN-Docking*. Coordinates (x, y, z) belong to the mass center of the ligand. (a, bi, cj, dk) represents the rotational quaternions and their norm. (a_0, a_1) indicates the possible action to be chosen, that is, moving forward along axis x or backward. Finally, es, ww, 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 computed by METADOCK.

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 METADOCK SF, which implicitly takes into account the structure of the host. Those two functions–reward and SF–are described in the lines below.

Moreover, METADOCK is conceived as the RL environment ε. This software has recently been proposed as an efficient heuristic-based software framework that makes affordable the study of protein-ligand interactions that occur during the ligand-host binding process. Other alternatives such as AutoDock Vina or Smina were considered but METADOCK stands out for being computationally

faster, which is an important aspect with regard to DRL training. In particular, METADOCK can apply translations and rotations to the ligand in the Euclidean space, and report the quality of the movement taken by using a force-field-based SF. This function involves the calculation of three major terms, as shown in Equation 3: (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.

$$\sum_{i=0}^{n}\sum_{j=0}^{m} k\left(\frac{q_i\ q_j}{r_{ij}}\right) + \sum_{i=0}^{n}\sum_{j=0}^{m} 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right) + \sum_{i=0}^{n}\sum_{j=0}^{m}\left(cos\theta_{ij}\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) + sin\theta_{ij}\ 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right)\right) \tag{3}$$

Algorithm 1 briefly describes how the score is computed by METADOCK given a particular position of the ligand.

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×ϵ×(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
```

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. However, unlike other settings such as the Atari videogames that DQN was orginally designed for, this score is not 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. Therefore, this score turns out to be too noisy to be employed as a reward signal and it does not favor convergence of the RL algorithm. As a workaround, the original SF 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 14 previous versions of the reward function were tested until the final candidate was selected.

Additionally, the value function in *DQN-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 replay dataset. This simpler architecture is chosen over other more compelling alternatives because of the relative simplicity of the inputs. In particular, CNNs and RNNs are discarded since *DQN-Docking* does not use 2D/3D grids or molecular sequences as input data, unlike the works analyzed in Section 3. 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 spe-

Algorithm 2 Reward function in *DQN-Docking*

Require: *es*: electrostatic term; *ww*: Wan der waals forces; *hb*: hydrogen bonds; *sc*: overall score from METADOCK 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**
        $sc_r = \eta$
      end if
    end if
  end if
end if
reward = add(es_r, ww_r, hb_r, sc_r)

cific hyper-parameters of the ANN model are listed in Table 1.

Another important aspect in RL refers to stop conditions. Docking has been conceived in this work as a finite-horizon, episodic task. METADOCK takes care of shifting the agent in the three-dimensional space and subsequently computing the SF but does not specifies when a given episode is over. As a consequence, two stop conditions are manually added. First, a maximum number of time-steps is set as in any RL episodic task (see Table 1). Second, sometimes the ligand may deviate and get way 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 10 following time-steps. In practice, this condition limits the exploration area of the agent around the host. 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 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 this work, 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.

5. Evaluation methodology

Once it is explained how the new docking method is implemented, next it is introduced the followed methodology to evaluate the proposed solution. As for the involved molecules, the evaluation of *DQN-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 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 forward and backward along the axis that cross through the inner hole of the cyclodextrin and its optimal spot. 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 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 experiment is 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, and a NVIDIA GeForce GTX 780 GPU (Kepler). OpenAI Baselines 0.1.5 is used to deploy the DQN algorithm. This library is based on Tensorflow and Keras frameworks to design and train ANNs. Specifically, TensorFlow 1.7.0 and Keras 2.1.5 are used for the experiment.

6. Results and discussion

First of all, a manual hyperparameter tuning [9] focused on execution time is carried out in order to select the optimal combination of DRL hyperparameters

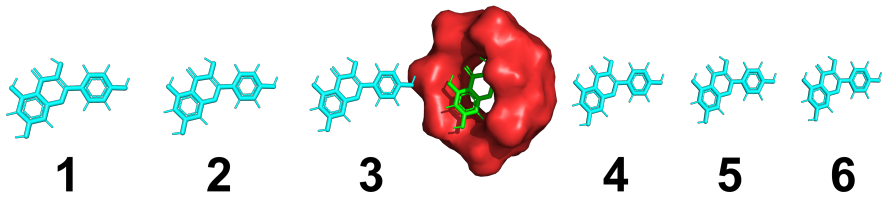

Figure 4: Evaluation methodology based on beta-cyclodextrin and kaempferol. The agent is independently trained from six different positions numbered from 1 to 6. The objective for the agent is to discover the optimal solution in the center of the cyclodextrin and stay oscillating around that spot. In 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.

and speed up training. The performed analysis encompasses more than 180 runs with different combinations of both Deep and Reinforcement Learning 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 maximum of global timesteps of the experiment directly affects the impact of the exploration fraction on execution time. Those interrelations are carefully taken into account 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 for initial position no. 3.

Figure 5 shows the hyperparameters that prove to have a deeper impact in

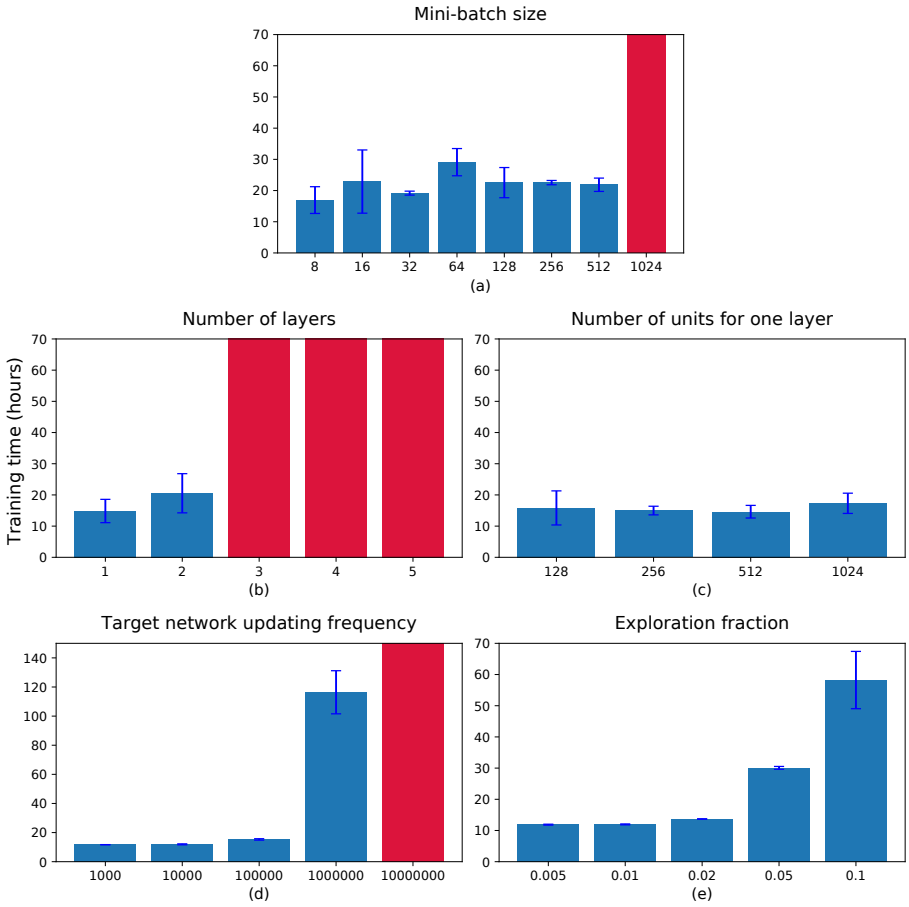

Figure 5: Result of hyperparameters analysis. Hyperparameters values are tested sequentially. Red bars with diagonal stripes indicate that the algorithm failed to converge. Error bars are based on confidence intervals with $\alpha = 0.05$. Note that the y-axis scale is the same for all the bar charts, ranging from 0 to 70, but for 5.d, which ranges from 0 to 150.

terms of computational efficiency. Overall, exploration fraction and mini-batch size are the most determining ones. With respect to the mini-batch size (bar plot 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 (5.b). The ANN with one layer seems to be the most suit-

Table 1: Values of the hyperparameters used in *DQN-Docking*

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	Angstroms 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

DL 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

able 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 (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 (5.d) is not specially important but values greater than 1 million make the algorithm unfeasible to converge. Moreover, the exploration fraction (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 1 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 6. It is calculated

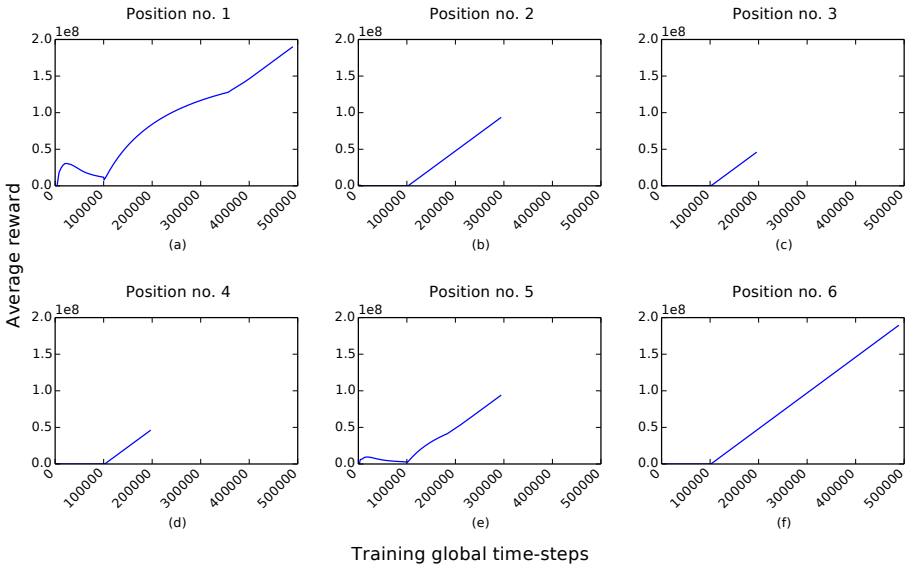

Figure 6: Average total reward per time-step during the training process. The average is calculated considering the previous 100 episodes. For the sake of comparison, both x and y axes share the same range of values in the six charts.

with respect to the previous 100 episodes. 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, the heatmap in Figure 7a shows the Root Mean Square Deviation (RMSD) in Angstroms between the last position in the episode of prediction and the optimal solution. This distance is calculated for each pair of training and prediction initial positions. The purpose is to demonstrate that

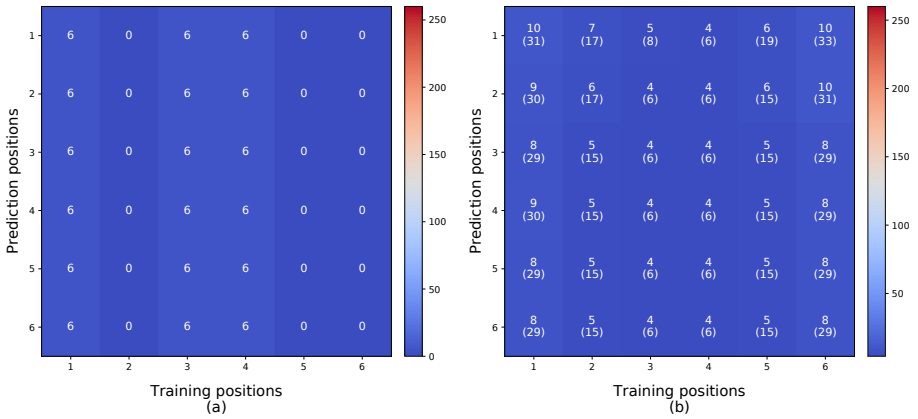

Figure 7: (**a**) RMSD between the last position in the episode of prediction and the optimal solution. The distance is computed for each pair of training and prediction initial positions. The maximum distance between the current position and the solution across the 36 pairs is 256 Å. (**b**) Average RMSD between current position and the optimal solution across the episode of prediction. The standard deviation is shown in parentheses.

the agent is able to find the optimal spot regardless of which was the initial position during training. 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, the average RMSD in Figure 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.

7. Conclusions and future work

Computational drug discovery is an amazing field with an incredible room for extension, innovation, and impact. The same is true for VS and docking, where the fastest methods are not currently able to process the largest biological databases in a reasonable time-frame. This limits its practical application in medical research. As a consequence, there is an appealing, urgent research gap to be filled. Instead of mainly relying on parallel programming schemes and more powerful hardware to accelerate the resolution of the PLDP problem, we took an alternative approach based on the latest advances in DRL. Particularly, we created a system with an embedded DQN 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. Results from both training and prediction phases demonstrate that once the agent has been properly trained it is able to find the solution where both molecules interact, irrespective of the original position it was trained from.

We do believe that this pioneering study is a valuable first milestone to generalize our findings to other ligand-host pairs and thus reach the ambitious goal of developing a faster, more accurate tool to solve the PLDP problem. By extending the present work, we hope to contribute to save tons of economic resources for the pharmaceutical industry and, what is more important, to save the life of those who urgently need the drugs to be develop.

Acknowledgements

This work was partially supported by the Fundación Séneca del Centro de Coordinación de la Investigación de la Región de Murcia under Projects 20813/PI/18, 20988/PI/18 and 20524/PDC/18, and by the Spanish Ministry of

Science, Innovation and Universities under grants TIN2016-78799-P (AEI/FEDER, UE) and CTQ2017-87974-R. The authors also thankfully acknowledge the e-infrastructure program of the Research Council of Norway, and the supercomputer center of UiT - the Arctic University of Norway.

References

[1] Kai Arulkumaran, Marc P Deisenroth, Miles Brundage, and Anil A Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[2] 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.

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

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

[5] Fahimeh Ghasemi, Alireza Mehridehnavi, Afshin Fassihi, and Horacio Pérez-Sánchez. Deep neural network in qsar studies using deep belief network. *Applied Soft Computing*, 62:251–258, 2018.

[6] Fahimeh Ghasemi, Alireza Mehridehnavi, Alfonso Pérez-Garrido, and Horacio Pérez-Sánchez. Neural network and deep-learning algorithms used in qsar studies: merits and drawbacks. *Drug Discovery Today*, 23(10):1784, 2018.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] William L Jorgensen. The Many Roles of Computation in Drug Discovery. *Science*, 303:1813–1818, 2004.

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

[20] Douglas B Kitchen, Hélène Decornez, John R Furr, and Jürgen Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. *Nature Reviews Drug Discovery*, 3(11):935–949, 2004.

[21] 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.

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

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

[24] Antonio Lavecchia. Deep learning in drug discovery: opportunities, challenges and future prospects. *Drug Discovery Today*, 24(10):2017–2032, 2019.

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

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

[27] 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.

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

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

[30] 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.

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

[32] 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.

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

[34] 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.

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

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

[37] 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.

[38] Matthew Ragoza, Lillian Turner, and David R Koes. Ligand pose optimization with atomic grid-based convolutional neural networks. *arXiv preprint arXiv:1710.07400*, 2017.

[39] 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.

[40] 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.

[41] 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.

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

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

[44] Marwin HS Segler, Mike Preuss, and Mark P Waller. Learning to plan chemical syntheses. *arXiv preprint arXiv:1708.04202*, 2017.

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

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

[54] Ana Tapia-Abellán, Diego Angosto-Bazarra, Helios Martínez-Banaclocha, Carlos de Torre-Minguela, Jose P Cerón-Carrasco, Horacio Pérez-Sánchez, Juan I Arostegui, and Pablo Pelegrin. Mcc950 closes the active conformation of nlrp3 to an inactive state. *Nature Chemical Biology*, 15(6):560, 2019.

[55] 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.

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

[57] 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.

[58] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

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

[60] 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.

[61] 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.

[62] Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2019.