Accelerating Drugs Discovery with Deep Reinforcement Learning: An Early Approach

Antonio Serrano*
Universidad Católica de Murcia (UCAM)
Murcia, Spain
aserrano7@ucam.edu

Baldomero Imbernón
Universidad Católica de Murcia (UCAM)
Murcia, Spain
bimbernon@ucam.edu

Horacio Pérez-Sánchez
Universidad Católica de Murcia (UCAM)
Murcia, Spain
hperez@ucam.edu

José M. Cecilia
Universidad Católica de Murcia (UCAM)
Murcia, Spain
jmcecilia@ucam.edu

Andrés Bueno-Crespo
Universidad Católica de Murcia (UCAM)
Murcia, Spain
abueno@ucam.edu

José L. Abellán
Universidad Católica de Murcia (UCAM)
Murcia, Spain
jlabellan@ucam.edu

ABSTRACT

Clinical research can be remarkably enhanced making use of virtual screening techniques that predicts how ligands interact with pharmacological receptors. This accelerates the long and expensive process of finding new drugs. Current methods often involve computer clusters that require high computational cost and several programming models. As an alternative to alleviate these problems, we developed DQN-Docking, a novel approach that takes advantage of the latest breakthroughs achieved in deep reinforcement learning by applying a Deep Q-Network to the context of protein-ligand docking prediction. The goal is to design a system able to "teach" the agent–the ligand–how to successfully couple with the receptor by an iterative trial-and-error process. A Deep Q-Network estimates the Q-value function to guide the agent to move in the environment, while METADOCK, a parallel metaheuristic schema for virtual screening methods, provides the resulting state of those movements and its corresponding docking score. Preliminary results seem promising but more research needs to be conducted to refine DQN-Docking and make it an effective alternative to solve the molecular Docking problem.

CCS CONCEPTS

• **Applied computing** → **Bioinformatics**; • **Computing methodologies** → **Reinforcement learning**; *Neural networks*;

KEYWORDS

Virtual Screening, Docking, Deep Reinforcement Learning, Deep Q-Network

*This is the corresponding author

ICPP '18 Comp, August 13–16, 2018, Eugene, OR, USA

ACM ISBN 978-1-4503-6523-9/18/08…$15.00
https://doi.org/10.1145/3229710.3229731

ACM Reference Format:
Antonio Serrano, Baldomero Imbernón, Horacio Pérez-Sánchez, José M. Cecilia, Andrés Bueno-Crespo, and José L. Abellán. 2018. Accelerating Drugs Discovery with Deep Reinforcement Learning: An Early Approach. In *ICPP '18 Comp: 47th International Conference on Parallel Processing Companion, August 13–16, 2018, Eugene, OR, USA,* Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3229710.3229731

1 INTRODUCTION

Traditional drug discovery processes usually take around one decade from initial hit discovery to drug approval [15]. This significant amount of time has been recently shortened by the use of Virtual Screening (VS) methods. Nowadays, among them, one of the most effective ones is Docking [23] to solve the so-called Protein-ligand Docking Prediction (PLDP) problem. Docking is a computational model that relies on rapid and accurate molecular simulation to recreate the atomic interactions among atoms that are involved in the drug discovery process. Hence, the usage of Docking methods not only leads to important reductions of research-to-market cycles but also to vast cost savings [21]. Nonetheless, molecular simulation at scale to cope with realistic problems in Docking demands large amount of computing power. So, widespread adoption of high-performance computing (e.g., massively parallel GPGPU platforms and parallel programming models such as CUDA or OpenCL) and novel algorithmic techniques based on Artificial Intelligence (AI) have been proved successful for contemporary Docking methods [6, 12, 33].

Simultaneously, the field of AI has grown exponentially in the last decade. That great momentum is mainly explained by the last successes achieved in Machine Learning, specially in Deep Learning (DL) [25]. DL is a family of algorithms based on learning data representations, whose most representative models are Neural Networks (NN). NN are computing systems vaguely inspired by the biological neural networks. A NN is made of many simple computing units (artificial neurons) organized in sequential layers [45]. So, the first layer receives the raw input information e.g. pixels from images. Each successive layer receives an output from the previous layer until the last layer, which produces the final output of the system. In this process, each artificial neuron computes the

weighted sum of its input and applies to the result of this calculation an (often non-linear) activation function (e.g. sigmoid or ReLU) to produce the final neuron's output. In classification tasks, for instance, neurons belonging to the first layers tend to specialize in low-level features (such as strokes, lines and other basic forms), while last-layers neurons highlight high-level aspects of the inputs, recognizable by humans, that are significant for discrimination (e.g. in face recognition, it would be the form of noses, eyes, oval faces, etc.). If a NN has many hidden layers, it is referred as Deep NN.

Most modern algorithms of DL have been with us since the mid-80s. However, over the last few years, the applications for DL have grown exponentially, mainly thanks to more powerful computing platforms such as Graphics Processing Units (GPUs) [53] or specific hardware accelerators such as the Google TPU [22], mature software packages and architectures, new algorithmic techniques, and strong financial support [28]. As a consequence, DL boomed in many domains such as image recognition [24, 54], self-driving cars [4], speech recognition [42], and machine translation [51], to name but a few. They have been also successfully applied in drug discovery [6, 12, 33], including application such as QSAR [32], genomics [1, 61], and VS [8, 58].

Another well-known sub-discipline in Machine Learning is Reinforcement Learning (RL). RL is about teaching an agent how to interact with the environment, optimizing a policy by trial and error in sequential decision making problems [52]. This family of algorithms has also taken advantage of the advent of DL, by integrating neural networks in some of its components [28] i.e. the policy, the value function, or the transition model–a more detailed explanation of RL will be exposed in Section 2.2. That integration dates back to a long time ago but there have been two important milestones–Deep Q-network (DQN) [35, 36] and AlphaGo [47, 48]. These two algorithms gave rise to a new promising approach called Deep Reinforcement Learning (DRL). In a nutshell, DRL is a relatively new family of algorithms that make use of DL in a broader framework of RL and is poised to revolutionize the field of AI in the coming years [2].

Some of the most remarkable achievements in DRL took place in boardgames–such as the aforementioned AlphaGo–and videogames–not only those of Atari 2600 [36] but also more realistic 3D environments like Doom [60] and Minecraft [55]. These applications were expected since games have been natural testing labs to AI for decades. But DRL has been recently also applied to other areas such as robotics [14, 26], natural language processing [27, 38], or healthcare [31, 39]. This new approach was also used in drug discovery recently, namely in *de novo* molecular design [40].

In spite of the proven benefits of VS, it naturally comes with its own downsides. The computational processes of VS filter a set of chemical compounds, known as *ligands*, and deduce which of them are capable of interacting with a given protein, called *receptor*, through energy calculations. Each of these ligands can be evaluated in millions of different positions with respect to the protein to find the best possible coupling location. Therefore, this Docking process is very demanding computationally speaking even for GPU-based methods. Considering this disadvantage of VS, on the one hand, and the latest breakthroughs in DRL, on the other hand, we propose *DQN-Docking*, a novel approach based on the original DQN algorithm. The goal is to accelerate the Docking process between a given ligand and the corresponding receptor molecule.

To the best of our knowledge, this is the first work that proposes to apply DRL to solve the PLDP problem. In particular, we present our early experiences testing DQN-Docking in one particular instance from Worldwide Protein Data Bank [3]–the receptor-ligand pair known as 2BSM. Our main goal is to teach the ligand to better approach and explore the receptor surface to discover the crystallographic solution (the one that makes both molecules interact with one another), or at least positions with similar scores as those obtained with state-of-the-art Monte Carlo optimization methods. Furthermore, the ultimate goal would be to make DQN-Docking scalable to any other scenario beyond 2BSM. Thus, we hope to contribute to literature by finding better solutions in terms of scoring, by reducing the computational cost once the NN is already trained, and by building a base to develop a more ambitious, scalable algorithm applicable to any other ligand-receptor setting in the near future.

The rest of this paper is organized as follows. First, a theoretical background is provided in Section 2 regarding VS and DQN. In particular, we succinctly review the generalities of VS. We also describe the general problem of RL and the approach of DQN to estimate the Q-value function. Second, Section 3 explains how DQN is implemented to solve the PLDP problem. Thus, it identifies the main components of RL in this context, and describes the environment of METADOCK, how the scoring function is computed, the NN architecture employed, and the basic rules to be followed by the agent. Third, Section 4 informs about the specific hardware and software used in the experiment, and the values of the RL and DL hyperparameters. As expected, it also reports our early results obtained at this stage. Finally, Section 5 concisely explains the conclusions of this work and describes the corresponding limitations and future work.

2 BACKGROUND

2.1 Virtual Screening

VS methods are computational techniques that analyze large libraries of small molecules with less than 200 atoms, called ligands. The goal is to find those able to bind to a second, bigger molecule involved in a given disease (see Figure 1). This other molecule, also known as the target, is typically a protein receptor or enzyme [41]. These libraries of chemical compounds may contain millions of ligands [19]. So it is normally assumed that the larger and more chemically diverse the database, the higher chances of discovering new drugs. This can be explained by the fact that analyzing larger databases, exponentially increases the chances of generating hits. However, current VS methods, such as Docking [23], fail to make good toxicity and activity predictions since they are constrained by their access to computational resources and the theory level used in their scoring functions. In fact, the fastest VS methods cannot process large biological databases in reasonable time.

The use of high performance computing in order to enhance VS methods is therefore necessary to fulfill pharmaceutical industry expectations. Consequently, a lot of research is being carried out in this regard. VS Docking based methods such as Autodock [37], Autodock VINA [57], Glide [11], LeadFinder [50], SurFlex [20],

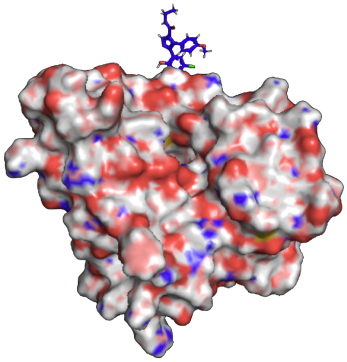

Figure 1: Ligand and receptor in the 2BSM setting. The ligand (in dark blue) is shown at the top of the protein.

ICM+ [49], FMD [7] or DOCK [9] use multithreading programming at the node level in order to leverage multicore architectures. Some of them even distribute their computations among the CPUs of several nodes by means of the Message Passing Interface (MPI) library. However, we are currently witnessing a steady transition to heterogeneous computing systems [56], with heterogeneity representing systems where nodes combine traditional multicore architectures (CPUs) with accelerators such as Graphics Processing Units (GPUs). Applications such as BUDE [34], AMBER [43], BINDSURF [44] or METADOCK [18] use GPUs to overcome this problem by dividing the whole protein surface into independent regions or spots. In addition, METADOCK uses a parameterized metaheuristic scheme applied to the PLDP problem, where several heuristic strategies can be applied. For the selected heuristic, METADOCK evaluates the ligand in millions of positions by varying translational and rotational degrees of freedom around the surface of the receptor. It computes a scoring function to measure the goodness of each position relying on GPU to speed up the process. This scheme differs METADOCK from traditional models applied to perform virtual screening processes, such as the Monte Carlo algorithm. However, heterogeneity may limit system growth as it can no longer be addressed in an incremental way. Actually, several computational challenges come up with such heterogeneous systems [5], like scalability, programmability or data management, to mention just a few.

2.2 Deep Q-Network

As stated before, in RL an agent interacts with the environment trying to learn an optimal control policy by trial and error to maximize the sum of some kind of reward signal. The agent learns across a set of subsequent attempts or episodes made of several time-steps. As it is depicted in Figure 2, at a given time-step t, the agent observes a state s_t, takes an action a_t, gets a reward r_t from the environment, and transitions to a new state s_{t+1}. Note that, unlike supervised learning, there is no explicit dataset. The environment generates the data and responds to the actions of the agent.

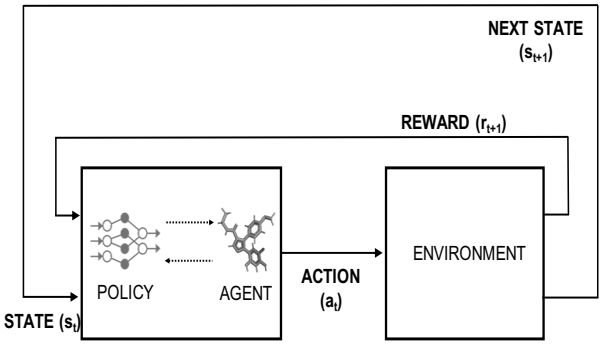

Figure 2: DQN basic operation.

In order to select the best action in a given time-step that maximizes the reward, a Q-learning approach [59] is taken. This value-based, model-free algorithm estimates the utility values of executing an action by continuously updating the Q-values using the following rule based on the Bellman equation:

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

Where $Q(s,a)$ is the expected utility of taking action a in the state s, α is the learning rate (i.e. how fast are we approaching the goal), and γ is a discount factor (normally close to 1) that is used under the assumption that an uncertain future reward worth less than the reward at present. In other words, the quality of taking an action a given a particular state s is the sum of the immediate reward r plus the discounted value of the next state s'. Note that Q-learning is off-policy because the next action a' is selected to maximize the next state s' Q-value instead of following the current policy as in on-policy methods. In particular, DQN follows an ϵ-greedy strategy to manage the exploration/exploitation trade-off. So, at first the agent takes random actions to explore the environment. As the NN iteratively updates the weights and its Q-values become more reliable, the agent begins to take actions based on those predicted Q-values.

Since NN can be used as function approximators, we can use them to learn a non-linear function for the Q-values of $Q(s,a|\theta)$, being θ the weights from the NN that parametrize those Q-values. The NN weights are updated iteratively during a training process. In every of those iterations, the goal is to minimize the next loss function:

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

Particularly, weights are updated in the next iteration $i+1$ based on the well-known backpropagation algorithm by computing $\theta_{i+1} = \theta_i - \alpha \nabla_\theta L(\theta_i)$. We chose the RMSprop as update rule for this work, following [35], although other options like Adam [29] could have been applied as well. Just as in supervised learning, the loss is the difference between the target $r_t + \gamma \max_{a'} \hat{Q}(s',a'|\theta_i^-)$, also known as Q-learning target or target network, and the Q-network $Q(s,a|\theta_i)$–i.e. the predicted values by the NN. Those Q-learning targets are updated every C iterations.

For i in step:
take a

The application of NNs as approximators of the Q-value function was an unsolved research challenge for many years. The key contributions made by [36] are the use of an experience replay dataset [30], the aforementioned target Q-network to compute the loss, and reward clipping. First, the experience replay dataset is used to store a fixed number of "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 memories are uniformly sampled from the dataset in minibatches to train the NN. The addition of those memories helps to break the correlation between samples from subsequent time-steps, and therefore facilitates convergence. Second, keeping the target network frozen avoid oscillations and also helps to break correlations between that target network and the Q-network. Finally, as to the reward clipping, it is employed to normalize the network adaptively to a more sensible range providing more robust gradients.

3 DQN-DOCKING

Docking is very intensive computationally speaking since it is considered a ***NP-complete*** problem [46]. METADOCK [18] 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-receptor binding process. 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 scoring function. This function involves the calculation of three major terms, as shown in Equation 1: (1) electrostatic interactions [13]; (2) the potential of Lennard-Jones [16] as a mathematical model to solve Van der Waals' forces; and (3) the hydrogen bonds term [10]. In addition, Algorithm 1 briefly describes how the score is computed given a particular position of the ligand.

$$\sum_{i=0}^{n}\sum_{j=0}^{m} k\left(\frac{q_i\, q_j}{r_{ij}}\right) + \sum_{i=0}^{n}\sum_{j=0}^{m} 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right) + \sum_{i=0}^{n}\sum_{j=0}^{m} cos\theta_{ij}\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) + sin\theta_{ij}\; 4\epsilon_{ij}\left(\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right) \quad (1)$$

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

```
for i=1 to N_CONFORMATION do
  for j=1 to N_ATOMS_RECEPTOR 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
```

In this paper, we propose DRL as a novel approach for efficient computing of Docking method. Given the aforementioned explanation of METADOCK, it is not difficult to identify the major components of DRL that are present in the context of Docking: (1) the ligand, which plays the role of the agent in DRL; (2) the policy that will be represented by a NN estimating the Q-values; (3) the possible actions (translations and rotations) that the agent can take; (4) the environment ε represented by METADOCK; (5) the state, which corresponds to the internal state of METADOCK depicting the exact positions of ligand and receptor; (6) the next state given by METADOCK once the action has been taken; and (7) the reward that will be represented by the normalized change in the score given by the scoring function computed by METADOCK.

As exposed in Section 2.2, we chose DQN as the target implementation of DRL for Docking. That is why we name our proposal as *DQN-Docking*. However, unlike the original work of DQN, the Docking task is not intended to be completed by a human being, so there is no obligation to reproduce those human-alike conditions working with images as raw inputs. Instead, the internal state of METADOCK depicting the exact positions of ligand and receptor is given as raw input to the NN. In other words, we are facing a Markov Decision Process (MDP) problem, a particularization of the Partially Observed Markov Decision Process (POMDP) setting.

A pseudocode of DQN-Docking is detailed in Algorithm 2. Note that is basically a duplicate of the original DQN highlighting the interactions between DQN and METADOCK. At each time-step the agent takes an action a_t from the set of possible actions, $A = \{1, \ldots, K\}$. For this problem, we consider a set of 12 possible actions to be taken by the ligand, including shifting and rotating forwards/backwards in the three spatial axes. The action is passed to METADOCK, which computes the new position (next state) of the ligand and its corresponding score. The states are vectors $x_t \in R^d$ representing the position of the atoms of the ligand and receptor and their respective bonds.

Besides the raw inputs, there is another important difference between the Atari setting and the PLDP problem concerning the score. In those video games, the score is cumulative, while in the Docking setting the scoring function from METADOCK represents the quality of the position of the ligand in absolute terms. In addition, the game score is always positive and usually increases slightly from time-step to time-step. The score in METADOCK, however, is negative most of the time and can drop sharply if (1) the two atoms with positive charge from the ligand and receptor respectively get too close (*electrostatic repulsion*); or (2) the ligand overlaps the receptor (*steric repulsion*). Hence the range of the scoring function goes from big negative numbers (e.g. -4.5e+21) to 500 at most, depending on the molecules involved. Moreover, we follow the same criteria as the original DQN algorithm to input the score in our DQN implementation. First, the score is clipped between -1 and 1 to avoid pronounced changes. Second, we keep fixed all the positive rewards to be 1 and all the negative rewards to be -1, while unchanged rewards are set to 0. And third, the score is not directly taken as the reward but the change in that score.

With respect to the NN architecture, since the inputs in our case are not pixels as in the Atari video-game, similarly to [29], we opt for a standard feed-forward NN–also known as Multilayer

Algorithm 2 DQN-Docking

Initialize replay memory database D to capacity N.
Initialize action-value function Q (represented by the Q-network) with random weights.
Initialize target action-value function $\hat{Q}$ (the Q-learning targets) with weights $\theta^- = \theta$.
for episode = 1, M **do**
  Initialize state s_1 taken from METADOCK.
  for time-step = 1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \underset{a}{argmax}\, Q(s_t, a|\theta)$.
    Execute action a_t in METADOCK and retrieve reward r_t and next state s_{t+1}.
    Store transition $(s_t, a_t, r_t, s_{t+1}, terminal)$ in D.
    Sample random minibatch of transitions $(s_t, a_t, r_t, s_{t+1}, terminal)$ from D.
    Compute Q-learning target y_j:

$$y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max\limits_{a'} \hat{Q}(s_{j+1}, a'|\theta^-) & \text{for non-terminal } s_{j+1}. \end{cases}$$

    Perform gradient descent step on $(y_j - Q(s_j, a_j|\theta))^2$ with respect to the network parameters θ.
    Every C steps update $\hat{Q} = Q$.
  end for
end for

Perceptron (MLP). This NN is made of two layers whose size is equal to those elements in the state vector that really changes over each iteration, that is, the position of the atoms of the ligand. The specific hyper-parameters of our NN model are listed in Table 1.

Another important decision we must make in RL-based methods is defining the "game rules". In the context of DQN for training an artificial agent to play Atari video games, the emulator obviously includes the rules of those games. Therefore, the agent implicitly knows when to stop the episode, which shorten its duration and in turn speeds up training. Nevertheless, METADOCK does not include those stop conditions, so we had to add two of them manually. First, we restricted the movement area of the ligand to an additional third with respect to the euclidean distance between the mass centers of receptor and ligand at the initial state. If the ligand goes beyond that limit, the episode terminates immediately. Second, we observe that the ligand often goes through the receptor in such a way that the atoms of both molecules overlap. Although this behavior is allowed to let the agent exploring the internal recesses of the receptor, going deeper into the molecule beyond those nooks only makes the scoring function to dramatically decrease to big negative scores. That is, the desired behavior for the ligand is to go over the surface and do not waste too much time in the inner area of the protein. So again, to accelerate the learning process, DQN-Docking is designed to terminate the episode if the scores from 20 subsequent time-steps are smaller than a given threshold based on the scoring function. Specifically, we empirically determine that threshold to be -100,000.

Finally, Figure 3 shows the initial position of the ligand in blue (A) far away from the receptor in the 2BSM scenario. The ligand in green (B) shows the crystallographic ligand (i.e. the solution) located in a particular recess of the receptor. The goal is to "teach" the ligand to find that crystallographic spot by an iterative trial-and-error process. It should be noted that for the 2BSM setting, the crystallographic solution is known and unique although there exists other ligand-receptor pairs where the crystallographic spot is unknown beforehand and/or there are several of them at the same time.

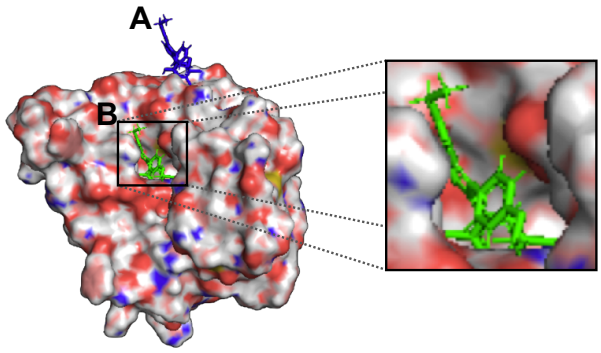

Figure 3: Initial (A) and crystallographic (B) positions of the ligand in the 2BSM setting. Crystallographic ligand is zoomed for clarity sake.

4 EXPERIMENTAL RESULTS

We perform our experiments 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). With respect to software, we used TensorFlow 1.7.0 and Keras 2.1.5 for NN designed and training. Table 1 shows the main hyper-parameters used in the learning process for both RL and DL frameworks. Some values are determined by the definition of the problem itself such as the state and action spaces, the shifting length, or the angle of rotation. Some others are set empirically like the update rate of the target network or the activation function of the neurons, or following previous works like the NN update rule [36] or the size of hidden layers [29].

As preliminary experimental results, we run DQN-Docking with the receptor-ligand pair 2BSM for a number of 1,800 episodes and analyze the quality of the training process. Accordingly, we take the estimated action-value function Q as the evaluation metric (see Figure 4), following [36]. This measure gives us an estimate of how much discounted reward the agent can obtained by following a certain policy from any given state. More specifically, we set a fixed number of time-steps before learning starts (see Table 1) and then track the average maximum predicted Q for each time-step from then on. As the NN learns how to map states to Q-values more accurately, the average maximum predicted Q value across episode should gradually increase.

In our case, the average Q-values per episode increase until reaching the level of 35,000 around episode 500, as expected. But then start to gradually decline until 27,000 in episode 1,800. Thus, we

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

RL hyperparameters		
Hyperparameter	Value	Description
Number of episodes M	1,800	Number of episodes to be completed along the simulation
Maximum time-steps limit T	1,000	Maximum time-steps limit per episode
State space	16,599	Real numbers needed to represent a particular state
Action space	12	Real numbers needed to represent the possible actions to be taken by the agent
Shifting length per step	0.1	Nanometers 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
Initial exploration steps	20,000	Number of initial steps where the agent only takes random action to explore the environment
ϵ initial value	1	Initial value of ϵ (if ϵ=1, then 100% actions are random at the beginning of training)
ϵ final value	0.05	Final value of ϵ (ϵ=0.1 means that 10% actions are random after the training process)
ϵ decay	4.5e-5	Decrease rate of ϵ per time-step to handle exploration/exploitation transition
γ discount rate	0.99	Discount rate for future rewards
Experience replay pool size N	400,000	Number of memories (s_t, a_t, r_{t+1}, s_{t+1}, terminal) to be stored to perform experience replay
Learning start	10,000	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

DL hyperparameters		
Hyperparameter	Value	Description
Number of hidden layers	2	The number of hidden layers between input and output layers
Hidden layer size	135	45×3 atoms of the ligand
Activation function	ReLU	Activation function used by hidden units to decide whether they should be activated or not
Update rule	RMSprop	The parameter update rule used by the optimizer
Learning rate	0.00025	Learning rate used by the optimizer
Minibatch size	32	Number of training examples per update

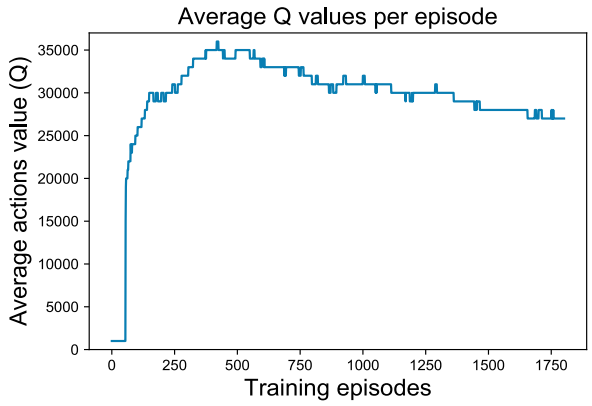

Figure 4: Training curve tracking the average predicted action-value.

cannot assure convergence. As explained in Section 5, the communication between DQN-Docking and the environment is relatively slow at this point of our research. This decelerates training, so we do not have enough episodes to really know if the algorithm will eventually converge to the optimal solution. On the positive side, we identify the main components of RL in the PLDP setting and embed METADOCK in a DQN system to be able to move the ligand based on Q-values predictions. But we need further exploration to fully guarantee convergence of DQN-Docking and achieve the goals stated in Section 1.

5 CONCLUSIONS AND FUTURE WORK

DQN is a cutting edge algorithm of highly proven convergence. It successfully solved the application of a NN as an approximator of the Q-value function in the RL context, allowing AI to solve problems that were consider intractable until recently. Furthermore, and what really many experts consider to be breakthrough results in the AI field, the same NN is able to play a great range of different videogames without making any particular modification for each of them. Thus, it is considered as an outstanding milestone in the long-dreamed quest of Artificial General Intelligence. In this work we tried to applied the mentioned algorithm to a new setting that could accelerate the drug discovery process, saving time and tones of economic resources, and hopefully the life of those people urgently in need of treatment.

We are currently in the middle of that process and the results seem promising. But the current implementation of DQN-Docking has several limitations. First, the communication between the algorithm and METADOCK entails to write two separate files in disk with the new state and the score respectively and then DQN-Docking reads those files to store memories in the experience replay dataset. We are refining the process to get a much faster RAM-based communication. Second, DQN-Docking was tested only with the receptor-ligand pair known as 2BSM. Although this is a real example of PLDP, the receptor protein is relatively small (3,264 atoms). The drawback of using internal states from METADOCK instead of

images is that the input size grows exponentially according to the number of atoms that define the molecules involved in Docking. It remains to be seen if the algorithm could be scale to bigger, more complex proteins without increasing the NN parameters excessively. In this regard, this work could be extended by substituting those internal states by a stack of receptor-ligand images and then use a convolutional NN instead of a MLP. Third, we worked with a fixed ligand. A more real setting would be working with flexible ligands able to rotate in certain flexible bonds. But at the same time it would add even more possible actions for the agent. For instance, in the 2BSM context, the ligand can fold in 6 bonds, so that would make a total of 18 possible actions. Fourth, up to now we applied the standard DQN but there exists new versions of this algorithm with their own pros and cons, such as DDQN, distributional DQN, dueling DDQN, etc. [17] It is worth to explore if those alternatives could be more suitable to the PLDP setting.

ACKNOWLEDGMENTS

This work is supported by the Spanish MINECO under grants TIN2016-78799-P and CTQ2017-87974-R (AEI/FEDER, UE).

REFERENCES

[1] Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey. 2015. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. *Nature biotechnology* 33, 8 (2015), 831.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866* (2017).

[3] Helen Berman, Kim Henrick, and Haruki Nakamura. 2003. Announcing the worldwide protein data bank. *Nature Structural and Molecular Biology* 10, 12 (2003), 980.

[4] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[5] Jesus Carretero and et al. 2014. Optimizations to enhance sustainability of MPI applications. In *Proceedings of the 21st European MPI Users' Group Meeting*. ACM, 145.

[6] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. 2018. The rise of deep learning in drug discovery. *Drug discovery today* (2018).

[7] Rafael Dolezal, Teodorico C Ramalho, Tanos CC França, and Kamil Kuca. 2015. Parallel Flexible Molecular Docking in Computational Chemistry on High Performance Computing Clusters. In *Computational Collective Intelligence*. Springer, 418–427.

[8] Dumitru Erhan, Pierre-Jean L'Heureux, Shi Yi Yue, and Yoshua Bengio. 2006. Collaborative filtering on a family of biological targets. *Journal of chemical information and modeling* 46, 2 (2006), 626–635.

[9] Todd J. A. Ewing, Shingo Makino, A. Geoffrey Skillman, and Irwin D. Kuntz. 2001. DOCK 4.0: Search strategies for automated molecular docking of flexible molecule databases. *Journal of Computer-Aided Molecular Design* 15, 5 (2001), 411–428.

[10] Felcy Fabiola, Richard Bertram, Andrei Korostelev, and Michael S Chapman. 2002. An improved hydrogen bond potential: impact on medium resolution protein structures. *Protein Science* 11, 6 (2002), 1415–1423.

[11] Richard A Friesner and et al. 2004. Glide: A New Approach For Rapid, Accurate Docking and Scoring: Method and Assessment of Docking Accuracy. *Journal of Medicinal Chemistry* 47, 7 (2004), 1739–1749.

[12] Erik Gawehn, Jan A Hiss, and Gisbert Schneider. 2016. Deep learning in drug discovery. *Molecular informatics* 35, 1 (2016), 3–14.

[13] Michael K Gilson, Kim A Sharp, and Barry H Honig. 1988. Calculating the electrostatic potential of molecules in solution: method and error assessment. *Journal of computational chemistry* 9, 4 (1988), 327–335.

[14] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 3389–3396.

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

[16] Thomas A Halgren. 1996. Merck molecular force field. II. MMFF94 van der Waals and electrostatic parameters for intermolecular interactions. *Journal of Computational Chemistry* 17, 5-6 (1996), 520–552.

[17] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2017. Rainbow: Combining Improvements in Deep Reinforcement Learning. *arXiv preprint arXiv:1710.02298* (2017).

[18] Baldomero Imbernón, José M Cecilia, Horacio Pérez-Sánchez, and Domingo Giménez. 2017. METADOCK: A parallel metaheuristic schema for virtual screening methods. *The International Journal of High Performance Computing Applications* (2017), 1094342017697471.

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

[20] Ajay N Jain. 2003. Surflex: fully automatic flexible molecular docking using a molecular similarity-based search engine. *Journal of Medicinal Chemistry* 46, 4 (2003), 499–511.

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

[22] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3079856.3080246

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

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

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[26] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[27] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* (2016).

[28] Yuxi Li. 2017. Deep Reinforcement Learning: An Overview. *CoRR* abs/1701.07274 (2017). arXiv:1701.07274 http://arxiv.org/abs/1701.07274

[29] Yan Li, Kenneth Chang, Oceane Bel, Ethan L Miller, and Darrell DE Long. 2017. CAPES: unsupervised storage performance tuning using neural network-based deep reinforcement learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 42.

[30] Long-Ji Lin. 1993. *Reinforcement learning for robots using neural networks*. Technical Report. Carnegie-Mellon Univ Pittsburgh PA School of Computer Science.

[31] Ying Liu, Brent Logan, Ning Liu, Zhiyuan Xu, Jian Tang, and Yangzhi Wang. 2017. Deep Reinforcement Learning for Dynamic Treatment Regimes on Medical Registry Data. In *Healthcare Informatics (ICHI), 2017 IEEE International Conference on*. IEEE, 380–385.

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

[33] Polina Mamoshina, Armando Vieira, Evgeny Putin, and Alex Zhavoronkov. 2016. Applications of deep learning in biomedicine. *Molecular pharmaceutics* 13, 5 (2016), 1445–1454.

[34] Simon McIntosh-Smith, James Price, Richard B Sessions, and Amaurys A Ibarra. 2015. High performance in silico virtual drug screening on many-core processors. *The International Journal of High Performance Computing Applications* 29, 2 (2015), 119–134.

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

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[37] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson. 1998. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *Journal of Computational Chemistry* 19, 14 (1998), 1639–1662.

[38] Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941* (2015).

[39] Shamim Nemati, Mohammad M Ghassemi, and Gari D Clifford. 2016. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the*. IEEE, 2978–2981.

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

[41] Judith M Rollinger, Hermann Stuppner, and Thierry Langer. 2008. Virtual screening for the discovery of bioactive natural products. In *Natural Compounds as Drugs Volume I*. Springer, 211–249.

[42] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. 2013. Deep convolutional neural networks for LVCSR. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*. IEEE, 8614–8618.

[43] Romelia Salomon-Ferrer, Andreas W GoÌĹtz, Duncan Poole, Scott Le Grand, and Ross C Walker. 2013. Routine microsecond molecular dynamics simulations with AMBER on GPUs. 2. Explicit solvent particle mesh Ewald. *Journal of Chemical Theory and Computation* 9, 9 (2013), 3878–3888.

[44] Irene Sánchez-Linares, Horacio Pérez-Sánchez, José M Cecilia, and José M García. 2012. High-throughput parallel blind virtual screening using BINDSURF. *BMC Bioinformatics* 13, Suppl 14 (2012), S13.

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

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

[47] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[48] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354.

[49] P Smielewski, A Lavinio, I Timofeev, D Radolovich, I Perkes, JD Pickard, and M Czosnyka. 2008. ICM+, a flexible platform for investigations of cerebrospinal dynamics in clinical practice. In *Acta Neurochirurgica Supplements*. Springer, 145–151.

[50] Oleg V Stroganov, Fedor N Novikov, Viktor S Stroylov, Val Kulkov, and Ghermes G Chilov. 2008. Lead finder: an approach to improve accuracy of protein- ligand docking, binding energy estimation, and virtual screening. *Journal of Chemical Information and Modeling* 48, 12 (2008), 2371–2385.

[51] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[52] Richard S Sutton and Andrew G Barto. 2017. *Reinforcement learning: An introduction. Second edition. Completed draft.*

[53] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* 105, 12 (2017), 2295–2329.

[54] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1701–1708.

[55] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. 2017. A Deep Hierarchical Approach to Lifelong Learning in Minecraft.. In *AAAI*, Vol. 3. 6.

[56] Top500. 2017. Top500 supercomputer site. http://www.top500.org/. (accessed, April, 3th, 2017).

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

[58] Thomas Unterthiner, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter. 2014. Deep learning as an opportunity in virtual screening. In *Proceedings of the deep learning workshop at NIPS*, Vol. 27. 1–9.

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

[60] Yuxin Wu and Yuandong Tian. 2016. Training agent for first-person shooter game with actor-critic curriculum learning. (2016).

[61] Hui Y Xiong, Babak Alipanahi, Leo J Lee, Hannes Bretschneider, Daniele Merico, Ryan KC Yuen, Yimin Hua, Serge Gueroussov, Hamed S Najafabadi, Timothy R Hughes, et al. 2015. The human splicing code reveals new insights into the genetic determinants of disease. *Science* 347, 6218 (2015), 1254806.