

# Automatic Hit-to-Lead Optimization

Optimizing ligand binding affinity through the application of deep  
Q-learning to docking simulations

**R. Maccallum**

Thesis presented for the degree of  
Master of Science

## Abstract

The process of drug discovery broadly follows the sequence of high-throughput screening, optimisation, synthesis, testing, and finally clinical trials. This takes an average of 10 years and 2.6 billion US dollars, while recent surveys have shown that the productivity of drug discovery research is on the decline, with average failure rates for clinical trials approaching 90% in all disease categories.

This research investigates methods for accelerating this process through the use of machine learning algorithms capable of automatically designing novel ligands for biological targets. Recent work has demonstrated the viability of deep reinforcement learning, generative adversarial networks, Bayesian optimisation and autoencoders. This work builds on one of the state-of-the-art deep reinforcement learning molecular modification algorithms and through the integration of molecular docking simulations applies it to the targeted generation of novel antagonists for the adenosine triphosphate binding site of plasmodium falciparum phosphatidylinositol 4-kinase, an enzyme essential to the malaria parasite's development within an infected host.



Department of Computer Science  
University of Cape Town  
South Africa  
2021



# Contents

<b>Contents</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Motivation . . . . .	4
1.2 Research Questions . . . . .	5
1.3 Contributions . . . . .	6
1.4 Overview . . . . .	6
<b>2 Related Work</b>	<b>8</b>
2.1 Scientific Automation and Robotic Laboratories . . . . .	8
2.2 Problem Complexity . . . . .	9
2.3 Molecular Descriptors and Similarity . . . . .	11
2.4 QSAR and Combinatorial Synthesis Approaches . . . . .	14
2.5 Evolutionary Approaches . . . . .	15
2.6 Reinforcement Learning Approaches . . . . .	16
2.7 Autoencoding Approaches . . . . .	20
2.8 Discussion . . . . .	23
<b>3 Methods</b>	<b>26</b>
3.1 Agent . . . . .	27
3.2 Environment . . . . .	28
3.3 Rewards and Shaping . . . . .	29
3.4 Learning Algorithm . . . . .	30
3.5 Summary . . . . .	33
<b>4 Experiments and Results</b>	<b>34</b>
4.1 QED Optimization . . . . .	35
4.2 Ligand Generation . . . . .	36
4.2.1 Sparse Rewards from Docking Simulations . . . . .	36
4.2.2 Docking Simulations with Reward Shaping . . . . .	37
4.2.3 Generation from Reference Series . . . . .	37
4.2.4 Comparison with Existing Ligands . . . . .	38
4.3 Summary . . . . .	38
<b>5 Discussion</b>	<b>40</b>
<b>6 Conclusions</b>	<b>44</b>
6.1 Future Work . . . . .	45
<b>A Experimental Parameters</b>	<b>46</b>
A.1 Network Structure . . . . .	46
A.2 Environment Parameters . . . . .	46
A.3 Hyperparameters . . . . .	47
A.4 Discussion . . . . .	48
<b>Glossary</b>	<b>49</b>
<b>Acronyms</b>	<b>52</b>
<b>Bibliography</b>	<b>54</b>

# Chapter 1

## Introduction

Drug discovery is the identification of compounds, either biological or synthetic, which can be expected through theory or simulation to bind effectively to targets in the human body. Targets are, for example, [proteins](#), [DNA](#), [RNA](#), or other [endogenous macromolecules](#) which have been shown to regulate a given [pathology](#). Pathologies of interest include heart disease, cancer, autoimmune disorders, high cholesterol, bacterial or viral infection, mental disorders, pain, diabetes and so forth.

Drugs will usually affect their regulatory action by binding to a specific region of the target known as a [receptor](#). Binding to a [receptor](#) will trigger some change in the target which yields a cellular response and consequential physiological effects. The mechanism of binding between drug and [receptor](#) is usually a combination of [hydrogen bonding](#), [electrostatic attractions](#), and [van der Waals interactions](#) [[Bruice, 2011](#)].

Drug-[receptor](#) interactions are usually either [agonistic](#) or [antagonistic](#). [Agonists](#) will mimic the natural action of [endogenous](#) biomolecules and trigger the same response in a [receptor](#) while [antagonists](#) will block these [endogenous](#) molecules from docking without activating the [receptor](#). Anti-histamines are examples of [antagonistic](#) drugs, as they prevent the cold-like symptoms which are caused by histamine molecules docking with histamine [receptors](#) by blocking the [receptor](#) without triggering its response [[Bruice, 2011](#)].

To bind effectively here means to bind with a high [affinity](#) and [selectivity](#). [Binding affinity](#) (BA) in this context means that the drug ([ligand](#)) forms a strong bond with the target's [receptor](#) and is thus not easily displaced by other molecules with lower BA. [Selectivity](#) means that the drug has a low BA for [receptors](#) other than the target's, that is, drugs with a low [selectivity](#) for a given target receptor will bind to that target, but they will also bind to other [receptors](#), leading to unintended side effects. Maximizing [selectivity](#) thus minimizes negative side effects [[Bruice, 2011](#)].

Databases such as PubChem [[Kim et al., 2021](#)] and the Comparative Toxicogenomics database [[Davis et al., 2021](#)] maintain searchable catalogues of the currently known pathologies, as well as known interactions between [proteins](#), genes and chemicals for which there is evidence showing that they contribute to a given [pathology](#).

Pathologies are generally mitigated by regulating a target, and this is usually achieved by activating or deactivating the target, thus increasing or decreasing their associated cellular functions. [Activation](#) or [deactivation](#) requires that a [ligand](#) be of the correct size and shape and possess the correct functional groups to substitute for the [endogenous](#) molecules which would usually dock at the target's active site as part of normal biological processes.

Target regulation is the primary goal when searching for new drug candidates and a drug's potential is determined by its size, shape, and functional groups, as these will determine the molecular interactions (BA) between the drug and [receptor](#).

As of 2021, a few of the most popular drug-target databases were: DrugBank with 4,563 human

targets and 2,358 approved drugs [Wishart et al., 2018], The Therapeutic Target Database with 3,419 human targets and 2,649 approved drugs [Wang et al., 2019], and ChEMBL with 6,311 human targets and 2,715 approved drugs [Mendez et al., 2019].

In order to successfully regulate a target a drug must reach the intended region of the body in the right **concentration**. Therefore, besides target regulation there are many auxiliary properties which make a molecule an appealing drug candidate, such as the molecule’s **molecular weight**, **solubility** (water-octanol partition coefficient) and **synthetic accessibility (SA)**. A selection of relevant molecular properties are combined into a metric called the **quantitative estimate of drug-likeness (QED)**. A high QED indicates that a drug is likely well-received by the body and easy to synthesize.

Other examples of properties of interest besides **BA** and **selectivity** for a particular target **receptor** are, in the case of oral drugs, the drug’s resistance to the acidity of the stomach and enzymatic degradation by the liver [Bruice, 2011], a molecule’s **polarity**, which amongst other things, relates to the rate at which it crosses the blood-brain barrier, or it’s **hydrophobic** or **hydrophilic** character as this influences its ability to dock at certain **protein receptors**, or its **molecular polarizability**, the measure of how easily a molecule’s **polarity** is distorted by an external electric field, as this effects bonding interactions and therefore impacts a drug’s **pharmacokinetics** and **pharmacodynamics**. These are all possible optimization goals for lead compounds. An appealing candidate would have a **scaffold** with a high **BA** and **selectivity** for a given target’s **receptor** as well as optimal values for properties such as these.

The typical contemporary drug discovery and design process comprises three phases: hit screening, which is where libraries of molecules are filtered for collections of hit molecules which are likely to be applicable for the intended use case. Hit-to-lead optimization, which is where hits identified from screening are modified by chemists to become leads: molecules with high **BA** for the intended target. This is then followed by lead optimization which is where the optimized leads then have any number of auxiliary **physico-chemical** properties relevant to their intended application optimized accordingly.

The screening process is usually driven by a method called high-throughput screening, where libraries of drug candidates are run through increasingly selective simulations, or through classifiers trained on examples of known hit compounds, to identify potential leads for a given target. Once leads are identified they are then further screened through **bioassays**, where the synthesized compounds are added to **assay** plates containing the biological target and the resulting effect, if any, on the target is observed, usually via some form of **spectroscopy**. Despite how costly and time-consuming it is to prepare and analyse these **bioassays**, they are still not guaranteed to detect active leads since the *in vitro* and *in vivo* environments differ significantly, so a candidate which is active *in vitro* may prove to be less effective or even useless when tested *in vivo*. Conversely, in the case of prodrugs, which are drugs that become pharmacologically active due to a transformation which occurs within the body, it’s possible for drugs which are later revealed to be active *in vivo* to be inactive *in vitro* as they first need to be transformed into an active state within the body [Bruice, 2011]. This is an incredibly costly process, and the resulting leads are limited to subsets of already known molecules. However, the process is accelerated through the use of computational filters and robotic automation of the **bioassay** screening. The subsequent two phases on the other hand are still heavily reliant on the knowledge of expert chemists to select hit compounds and modify their structure for increased target **BA** and **physico-chemical** properties.

Therefore, the field of drug discovery and design has been the focus of research into its acceleration. It is not possible to manually specify all molecular variations that would satisfy all possible constraints, since the space of drug-like molecules is estimated to contain between  $10^{26}$  and  $10^{60}$  molecules (the estimated number of stars in the observable universe is approximately  $10^{24}$ ). And so **machine learning (ML)** is being actively explored as a way to chart a course through chemical space.

There has been substantial research into the generation of so-called median molecules, which are sets of novel **functional analogues** of a given set and are therefore expected to possess similar

chemistry. The term median is used to refer to how these novel molecules occupy the interstitial chemical space between known structures [Brown et al., 2004b] and **functional analogues** are two molecules which exhibit similar biochemical or physiological effects on the human body, but with potential variations in efficacy and side effects [Bruice, 2011]. This is intended to avoid the screening process by generating novel hits from existing hits. There has also been some research into generating novel molecules possessing **physico-chemical** properties within specified ranges in order to accelerate the lead optimization phase.

The ideal goal of applying **ML** to this problem is to be able to describe a given macromolecular target along with a desired role such as **agonist** or **antagonist**, to an algorithm, which would then design an optimal **ligand**, possessing maximal **BA** and **selectivity** for the given target, as well as optimal auxiliary properties such that the corresponding **pathology** would be successfully mitigated through its use. Or more simply, given a candidate lead compound or **chemotype scaffold**, a trained agent would apply a learned policy to optimize one or more properties subject to a set of constraints.

There is a panoply of ways in which **ML** has been applied to the problem of drug design and optimization. These include **evolutionary algorithms (EAs)** [Rupakheti et al., 2015, Ole Carstensen et al., 2011, Supady et al., 2015, Brown et al., 2004b, Brown et al., 2004a, Lameijer et al., 2005, Sheridan et al., 2000, Jones et al., 1997], **reinforcement learning (RL)** with adversarial training [You et al., 2018, Guimaraes et al., 2017] and **variational autoencoders (VAEs)** [Gómez-Bombarelli et al., 2018, Jin et al., 2018]. Within these methods, besides the expected differences in implementation of the learning methods, there is significant variation in the choice of how to represent molecules to the learning algorithm, and the specifics of how reward or fitness is measured.

The field is also adjacent to research in artificial life and synthetic biology where researchers are attempting to approach the problem of mitigating physiological pathologies through the use of techniques such as molecular programming, artificial cells, molecular cybernetics and evolutionary and self replicating molecular assemblers. In these fields the goal is to develop molecular machines capable of harnessing cellular processes to replicate how drugs regulate the *in vivo* environment as an alternative to traditional drug discovery and design [Dinh et al., 2015, Hagiya et al., 2016, Aubert-Kato et al., 2017, Hagiya et al., 2014, Cavalcanti et al., 2008b, Cavalcanti et al., 2008a, Duim and Otto, 2017, Liu et al., 2020].

## 1.1 Motivation

The drug discovery and design process can be partitioned into three primary categories: hit screening, which is where classes of drugs or **chemotypes** which share a similar molecular **scaffold** are identified as having a notable **BA** for a target **receptor**, hit-to-lead optimization, which is where hits obtained from the previous phase have their **BA** for the **receptor** increased through modification by experts, and lead optimization, which is where leads optimized to have high **BA** for the target have their other **physico-chemical** properties such as **solubility**, **selectivity**, **molecular polarizability**, **charge distribution**, **molecular weight** and others customized for their intended application. This period of discovery and design is then followed by synthesis, testing and the stages of clinical trials. This process takes on average 10 years and costs an average of 2.6 billion US dollars [DiMasi et al., 2016]. In addition to this, the productivity of drug discovery research has been shown to be on the decline, with average failure rates for clinical trials approaching 90% in all disease categories [Kadurin et al., 2017].

Therefore, the motivation for this research is to develop systems for accelerating these three phases of drug discovery and design so that this timeline can be reduced from years to weeks, yielding gains in drug efficacy and a concomitant improvement in the quality of life for those whose cumulative suffering would persist were the process not accelerated.

This goal is also positioned within the context of the burgeoning field of automated science and robotic laboratories [Burger et al., 2020] which shares a similar goal but across a broad spectrum of scientific fields. It is hoped that the contributions of this research will be relevant to the

development of systems capable of extending learning beyond simulations to robotic scientists which design their own experimental parameters and incorporate autonomously obtained experimental data to improve their designs and adjust their assumptions.

There has been significant progress towards automation of the hit-screening and lead optimization phases through the use of EAs, generative adversarial networks (GANs) [Guimaraes et al., 2017, You et al., 2018] and autoencoders (AEs) [Gómez-Bombarelli et al., 2018]. However, the primary goal for drug design is BA for the target macromolecule, and it is automation of this, the hit-to-lead optimization phase of the drug design process, which has been underexplored relative to the other two phases.

Therefore, this research seeks to apply ML to develop a generative algorithm capable of correlating features of molecular structure with BA for a specified binding site of a target molecule such that it is capable of generating ligands with maximal BA for the receptor site, and when given a molecular scaffold the algorithm will return a ligand with greater BA for the target.

Recent work in the field has demonstrated the efficacy of deep RL at lead optimization [Zhou et al., 2019] and thus this was the foundation for the methods implemented in this research.

## 1.2 Research Questions

This research focuses on the hit-to-lead phase of drug discovery and design. Therefore, we are concerned with algorithms capable of taking a molecular scaffold of a particular chemotype and returning a ligand optimized for binding with a target receptor. The specific questions which this research seeks to answer, are therefore:

1. Is RL through simulations of molecular docking a viable method of learning a policy for automatic hit-to-lead optimization?
2. What are the methodological requirements for implementing RL as a method of training an intelligent agent to autonomously generate ligands possessing high BA for a target macromolecule?
3. How do the ligands produced by a simulation-based generative RL algorithm compare to ligands currently available for a particular receptor?

The motivation for these questions is as follows. Given that the primary requisite of a drug candidate is BA for the target macromolecule, question 1 aims to elucidate the potential for RL as a method of generating potential ligands using only *a priori* calculations when the crystal structure of a macromolecule is available. This is a potential alternative to the numerous approaches which employ supervised learning that are consequently dependent on the availability of large amounts of labelled training data.

Given that the application of ML to the automatic design of drug candidates is still a nascent field, there are many open methodological questions, particularly for this case of RL in combination with *a priori* docking score (DS) calculations and crystal structure. Therefore, question 2 aims to illuminate the impact of various methodological considerations which pertain to the representation of molecular graph states, environment rewards, and chemical validity among others. These various methodological components would comprise the practical requirements for successfully applying RL in this context.

The ultimate goal of frameworks such as this is that the graphs produced by the agent can be taken with a reliable degree of certainty to demonstrate high BA when the ligand is synthesized and evaluated in a physical bioassay. The purpose of question 3 is therefore to compare the structures and DSs of ligands generated by the agent with those selected by chemists. The first axis of comparison would be the DSs of the artificially designed ligands and those selected by chemists. The second and more important comparison would be to assess the correlation between DS and real ligand potency.

## 1.3 Contributions

In addressing the questions stated above, this research applies RL to the problem of generative drug design, thus yielding a generative RL algorithm.

The research therefore contributes a framework for training an agent through simulations of molecular docking to both generate novel ligands for a macromolecular target, and to modify a given molecular scaffold such that the resulting ligand has improved BA, as estimated using docking score (DS) as a proxy, for the receptor site of a given target macromolecule whilst retaining a certain level of structural similarity its initial state.

The framework applies deep RL, specifically double-deep Q-learning [Mnih et al., 2013, Mnih et al., 2015], in an environment comprised of a ligand to be modified and a target macromolecule. The agent designs molecular graphs and employs the Morgan algorithm [Rogers and Hahn, 2010] to convert graphs to molecular fingerprints. The state of the environment is the current state of the molecular graph and possible actions are defined to be the graphs accessible from the current state. Given that the action-space changes for each state, the algorithm implements a method for learning from environments with inconsistent action spaces. Docking between the ligand designed by the agent and the target macromolecule is simulated using the Autodock-GPU [Santos-Martins et al., 2021] package, and the DS calculated by Autodock-GPU defines the reward received by the agent at each step of a design episode.

This contribution is useful for a number of reasons. It can accelerate the hit-to-lead phase of drug design by proposing optimized leads from hits identified during the screening phase. It can also be used to generate novel ligands from scratch or to repurpose ligands which were discarded during the drug design process due to having promising drug-like properties but inadequate BA for the intended receptor. These automatically generated leads may result from unintuitive changes due to the agent identifying unknown correlations between structural features and BA which could potentially elucidate unknown chemistry. The framework can be extended to the lead optimization phase by incorporating any number of physico-chemical properties in the environment rewards.

Such a system will hopefully assist chemists in their work, directing them towards optimal regions of chemical space and thus accelerating their research, reducing the time and cost required to bring new drugs to clinical trials and aiding in the identification of novel compounds for outstanding pathologies.

## 1.4 Overview

The rest of this thesis is divided into 5 chapters: Related Work (chapter 2), Methods (chapter 3), Experiments and Results (chapter 4), Discussion (chapter 5), Conclusions (chapter 6).

Chapter 2 reviews related work in ML applications to drug discovery, covering the popular approaches using EAs, AEs, Gaussian optimization, RL, adversarial training and Tanimoto similarity.

It also covers the different methods of evaluating the performance of generative algorithms in this context, these are the concepts of internal diversity, external similarity and validity of the generated sets, as well as the considerations that need to be made when choosing how to represent molecules in a learning framework.

Chapter 3 describes the three aspects of the methodology implemented in this research. The first is the design of the agent and the environment including the input representations used, how training episodes were defined, how the agent explored molecular space, how states of the environment were defined, the action space available to the agent, how docking between the ligands designed by the agent and the target receptor were simulated, and the reward function used to evaluate the environment's state. Second, this chapter details the mathematics behind the algorithm used to train the agent including the method for facilitating learning given the sparse-rewards nature of the environment. Finally, the chapter describes how the agent and environment

were applied to the task of hit-to-lead optimization.

Chapter 4 describes how the methods presented in the previous chapter were evaluated and presents the results of their evaluation. The evaluation presented in this chapter includes the optimization of QED when constructing a molecule starting from a single carbon atom, the maximization of DS for a target receptor when constructing ligands from a single carbon atom, the maximization of DS when beginning from a reference scaffold, and a comparison between ligands generated by the agent and currently available ligands for the target receptor.

Chapter 5 discusses the implications of the results from the previous chapter, and Chapter 6 draws conclusions from the discussion, reviews the contributions of this research and ends with suggestions for future work.



## Chapter 2

# Related Work

This chapter presents an overview of the problem and context for the methods implemented in this work by providing a summary of seminal contributions to the field. It ends with a discussion of this related work with reference to the hypotheses formulated in the introduction.

Section 2.1 introduces the concept and motivation for the robotic automation of science. Section 2.2 describes the asymptotic computational complexity of the problem of automatic drug design from an algorithmic perspective and why ML is a viable method as opposed to traditional approaches. Section 2.3 introduces the abstractions used when representing the problem to a machine. Section 2.4 summarizes the traditional approaches used by chemists, namely the rational construction of quantitative structure-activity relationships (QSARs) through combinatorial synthesis. Section 2.5 summarizes the first of the ML approaches: the application of EAs to explore the solution space through the stochastic recombination of molecular building blocks. Section 2.6 summarizes the application of RL algorithms to the problem to leverage gradients to more effectively search the solution space. Section 2.7 summarizes the application of AE algorithms to model the probability distributions which define the salient features of drug molecules and bias the generation of novel compounds to those from optimal regions of drug space. The chapter ends in section 2.8 with a discussion of the various approaches to the problem as motivation for and theoretical background to the methods in chapter 3 and the hypotheses formulated in section 1.2.

### 2.1 Scientific Automation and Robotic Laboratories

The integration of computation and robotic automation with scientific research has been consistently increasing since the advent of the field of computer science, and it would be hard to imagine conducting modern scientific research without some component of automated computation. This integration has usually been for the sake of automating processes such as screening, mixing, experiment regulation, and so forth. However, with the recent advances in ML there has been significant progress towards the development of autonomous robotic scientists, capable of observation, hypothesis generation, data collection and hypothesis testing. The goal for such robotic scientists is not to simply combinatorially evaluate a hypothesis for all possible permutations of variables, but to learn from their observations, construct new hypothesis and adjust the experiments accordingly.

For example, Adam [King et al., 2009] is a fully autonomous robotic laboratory designed to identify encoding genes for enzymes catalysing biochemical reactions thought to occur in yeast. The mobile robotic chemist [Burger et al., 2020] is a robot capable of autonomously navigating a laboratory and manipulating the experimental apparatus. The robot was designed to search for improved photocatalysts for hydrogen production from water. Instead of simply trying all combinations of reagents, the robot employed Bayesian search to incorporate its measurements and update its model of the search space and select the next set of experimental parameters.

Another example is ChemOS [Roch et al., 2018], a general purpose autonomous laboratory system capable of being applied to a wide range of research laboratories with autonomous hardware. The software provides, among other functionalities, a learning module which continuously proposes new experimental parameters to be executed on the laboratories robotic platforms, a communications module, to facilitate communication between the human scientist and the laboratory system, and an orchestration module for asynchronously integrating geographically distributed laboratories.

These are examples of a shift away from simply automating routine tasks to replicating the patterns of thinking which underpin the investigative reasoning that is central to empirical research, and leveraging this to accelerate scientific research in a novel way.

## 2.2 Problem Complexity

The problem of drug discovery or optimization can be reduced to: given a propositional formula (target *receptor*, *physico-chemical* property range) does a satisfying argument (*ligand*) exist? Such a problem is NP-Complete and the de facto contemporary approach in this context is brute-force search (HTS) through known subsets of chemical space [Ruddigkeit et al., 2012].

Linear search on an array of molecules in this way has  $\Theta(n)$  time complexity. Estimates calculate the upper limit of the number of possible molecules to be  $10^{80}$  and the number of small organic molecules, that is, those which would have particular relevance to drug discovery to be  $10^{60}$  [Ruddigkeit et al., 2012]. Assuming the best case scenario where one has access to a supercomputer with a throughput capacity on the order of petaFLOPS ( $10^{15}$ ), such as Oak Ridge National Lab’s Summit, and an ideal case where only one floating point operation is required to evaluate a known candidate, evaluation of  $10^{60}$  candidates would require  $10^{45}$  seconds, or  $\sim 3 \times 10^{37}$  years, where the age of the universe is  $2.3 \times 10^{33}$  years. And this is the ideal case where a candidate can be evaluated with only a single floating point operation, on one of the world’s fastest supercomputers, bearing in mind that comprehensive simulations are significantly more demanding.

A practical demonstration of this evaluated  $1.66 \times 10^{11}$  candidates from the GDB-17 database of small molecules against simple property calculations, a process which required 11 CPU-years (11 years on a gigaFLOP processor) [Ruddigkeit et al., 2012]. Extrapolating from this result with  $\Theta(n)$  time complexity, an evaluation of all  $10^{60}$  drug molecules on the same system would require  $\sim 7 \times 10^{50}$  years.

Even if one had a fast enough system capable of reducing the execution time of a brute-force approach to within a single lifetime, such an algorithm is still constrained to be presented with known compounds, whereas the ideal candidate may not be a known structure.

Clearly, a more effective combination of data structures and algorithms is needed to efficiently search the solution space of known candidates, as well as generating novel candidates. The question then is, how best to chart a path through chemical space to a region that is optimal for any given goal in drug design or materials science?

Given the intractability of conducting a simple linear search through all possible molecules, a traditional method of focusing the search is to build expert knowledge into an algorithm such that only relevant candidates are considered for screening.

This is traditionally done through the use of QSARs (fully described in section 2.4) to determine relationships between molecular fragments and *physico-chemical* properties or biological activity. One can then prioritize their search to combinatorial libraries of molecules containing the relevant fragments.

The problem with this approach is that while a particular fragment may be experimentally determined to contribute a particular *physico-chemical* property, a molecule’s chemistry is determined by all interactions between all of its constituents, and while the presence of a particular fragment may indicate the presence of a particular property, it does not predict the cumulative impact of the molecule’s total constitution, configuration and conformations.

Secondly, the rules which one might use to construct such a combinatorial library are incomplete, as evidenced by the fact that Lapinski’s rule of five, a set of five rules used as a rule of thumb to estimate the potential for a structure to be well received *in vivo*, is regularly violated by experimental measurements of known drugs.

So the prediction of physical properties and biological activity from structure requires a model not just of the chemistry of individual fragments, but of the effect that the interactions between those fragments will have on the respective properties and activity. Designing a molecule to meet a particular set of requirements is thus a multivariate optimization problem, something where ML methods such as EAs and artificial neural networks (ANNs) have been shown to excel [Coetzee and Nitschke, 2019, Huang et al., 2021, He et al., 2021, Jang et al., 2020].

Furthermore, by applying ML to drug design there is the potential for ANNs to learn previously unknown relationships between molecular structure and physico-chemical properties. Section 2.4 describes the traditional method whereby chemists will obtain such structure-activity relationships. However, the increasing preponderance of publicly available experimental data allows researchers to train ANNs to predict almost any physico-chemical property including quantum mechanical properties which can require days to simulate, and to account for the impact of the presence of other molecular fragments on those properties.

Regarding the generation of novel molecules, this array of predictive networks can then be used to provide multivariate reward signals to generative networks which learn to achieve their goals without any ontological understanding of chemistry or physics.

The increasing availability of large sets of experimental data for physico-chemical properties, BAs, therapeutic indications, reaction pathways and gene/protein regulatory interactions, makes possible entirely automated drug design pipelines where discriminative ANNs can learn to predict these properties directly from ground truth data without needing human-defined relationships. These discriminative networks can then constitute multivariate reward signals for training generative networks through RL. Thus, intelligent agents can learn to improve drug designs directly from experimental measurements without needing human intervention [Réda et al., 2020]. There’s also the potential for advancements in ML theory to improve these prospects.

Once a discriminative network has been trained to predict physico-chemical properties or BAs, interpretation and inversion techniques can be used to elucidate what structural features of a given model are responsible for the network’s response, potentially yielding novel insights about the relationship between structure and behaviour [Elton et al., 2019].

In cases where chemists and biochemists do not have reliable rules about how molecular structure will affect activity, such as how genetic variation impacts drug receptivity, but for which there is an ever-increasing amount of publicly available data from genome-wide association studies, there is the potential for ML to yield optimized designs prior to a rational understanding of the biological interactions.

That being said, the true potential of the applications of ML to drug design lie in self-contained and self-improving design loops where autonomous systems obtain experimental data from high-throughput screening of assays measuring, for example, BA and selectivity against a target. That experimental data is then used to train ANNs which can predict BA for the target. Those ANNs are then used as the reward signal for a generative network which learns to correlate structural features with high reward, designing a new set of candidate ligands in the process. This new set is then synthesized and evaluated in another high-throughput assay. Through this self-improving automated loop of (exploration) design → synthesis → measurement → learning, these systems have the potential to learn new relationships between molecular structure, physico-chemical properties, and target BA [Sanchez-Lengeling and Aspuru-Guzik, 2018].

This research focuses on methods for applying ML to the targeted generation of molecules and the potential of such intelligent systems as a foundation for the development of high-throughput autonomous robotic laboratories.

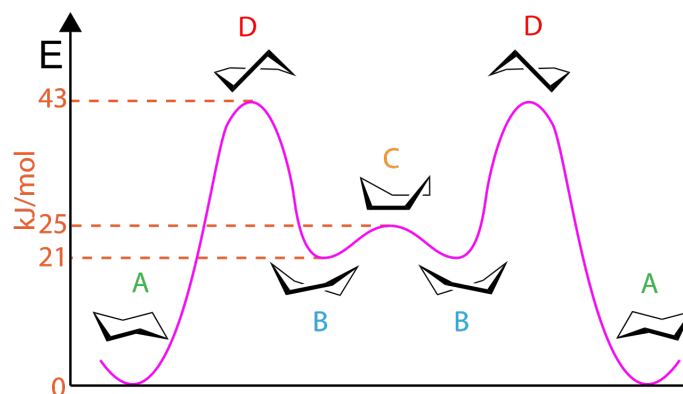


Figure 2.1: The conformations of cyclohexane [Wikipedia, 2021]. State A corresponds to the chair conformation and state C corresponds to the boat conformation. States B and D are transition states.

## 2.3 Molecular Descriptors and Similarity

A reliable rule of thumb in chemistry is that similar structures will yield similar behaviour [Nikolova and Jaworska, 2004]. Therefore, a significant topic in the field is how to best represent molecular structures so that similarity can be accurately measured [Nikolova and Jaworska, 2004].

When describing molecular structure one can define the constitution of a molecule, which refers to its two-dimensional structure as defined by the atoms it contains and how they are connected, which determines the presence of certain functional groups [Nikolova and Jaworska, 2004].

One can describe a molecule's configuration, that is to say, the three-dimensional arrangement of the molecule's constituents. Configuration can be described by the bond angles between atoms which are connected to at least two others [Nikolova and Jaworska, 2004].

Finally, one can describe a molecule's conformations, which is the set of thermodynamically stable spatial arrangements which the molecule will shift between as its bonds rotate due to the kinetic activity of its atoms [Nikolova and Jaworska, 2004]. A basic example of conformations are the boat and chair conformations of the cyclohexane molecule, shown in figure 2.1. These are two of cyclohexane's conformers, or shapes, which the molecule shifts between as its bonds rotate. The chair being the lower-energy conformer due to the molecule experiencing less steric hindrance (inter-atomic repulsion) and less torsional strain (inter-bond repulsion) in this shape, and the boat being the least stable conformer due to this shape possessing significant steric and torsional strain.

Together, a molecule's constitution, configuration and conformation will determine its activity and physical properties, as well as its shape, which is relevant for its interaction with receptors.

Theoretically, all relevant information about a molecule, including its constitution, configuration and conformation, is contained within its quantum mechanical wave-function, and so this would be the most informative representation to work with [Nikolova and Jaworska, 2004]. However, techniques for working with wave-function representations molecules are still underdeveloped and were therefore not explored in this project.

### Extended-Connectivity Fingerprints

Molecular fingerprints are a kind of topological representation which represent molecules as bit-strings. A particular kind of molecular fingerprint which is able to capture aspects of molecular activity relevant to drug activity are extended-connectivity fingerprints (ECFPs). These topological fingerprints were developed specifically for structure-activity modelling, and they are theoretically capable of representing an infinite number of molecular features, including stereochemical infor-

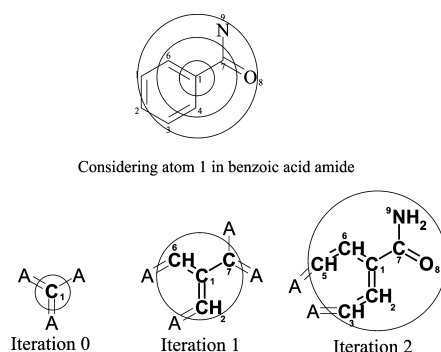


Figure 2.2: Visual representation of the **ECFP** algorithm. The circles represent that information that is incorporated into the identifier for atom  $C_1$  [Rogers and Hahn, 2010]. The radius of the circles increases with each iteration.

mation. These are sometimes referred to as Morgan Fingerprints as they are calculated using a variant of the Morgan algorithm; a method for solving the molecular isomorphism problem [Rogers and Hahn, 2010]. The **ECFP** algorithm is summarized as follows:

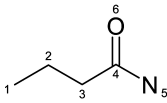
1. The molecule's atoms are assigned an integer identifier, this could, for example, be a series of atomic properties such as their atomic number, mass, charge, number of heavy neighbours, whether the atom is contained in a ring, and so on, or it could include information about the the atom's functional role in the molecule. These properties are defined by integer values which are then hashed into the atom's identifier.
2. The initial identifier for each atom is added to the fingerprint set.
3. Each atom adds its identifier, and those of its neighbours (bonded atoms) into an array. The identifiers of neighbours are added with their bond order: single (1), double (2), triple (3), or aromatic (4).
4. For each atom, this array is passed to a hash function which calculates a new identifier for the atom.
5. Each atom updates its identifier in the fingerprint set.
6. This loop of updating identifiers in the fingerprint set is iterated a set number of times.
7. After the set number of iterations, any duplicate identifiers are removed and the remainders define the molecule's **ECFP**.

This process computes an identifier for each atom which represents that atom's molecular neighbourhood and includes information not just about the atom's which are directly bonded to the given atom, but also the respective neighbours of the atom's neighbours. With every iteration an atom's identifier contains information about atoms further away. This is shown in figure 2.2.

When calculating a molecule's **ECFP**, the number of iterations determines the diameter of the circle of information that is incorporated into each atom's identifier, where the circle's diameter is defined in terms of the number of bonds it contains. For example, two iterations would lead to a circle with a diameter of four bonds being centred on each atom.

Fingerprints are named according to this diameter. So the fingerprint generated from two iterations would be an "**ECFP\_4**".

The result is a set of numerical identifiers for every atom in a molecule which represent the molecular environment in which the atom is situated.



> <ECFP_0>	> <ECFP_2>	> <ECFP_4>	> <ECFP_6>
734603939	734603939	734603939	734603939
1559650422	1559650422	1559650422	1559650422
-1100000244	-1100000244	-1100000244	-1100000244
1572579716	1572579716	1572579716	1572579716
-1074141656	-1074141656	-1074141656	-1074141656
	863188371	863188371	863188371
	-1793471910	-1793471910	-1793471910
	-1789102870	-1789102870	-1789102870
	-1708545601	-1708545601	-1708545601
	-932108170	-932108170	-932108170
	2099970318	2099970318	2099970318
		-87618679	-87618679
		1112638790	1112638790
		-627599602	-627599602

Figure 2.3: Example of the ECFP for butyramide calculated with different diameters [Rogers and Hahn, 2010].

The method is in some aspects similar to the message passing algorithm used in graph convolutional neural networks, and it produces a representation that is similar to graph embeddings.

An example of the ECFP calculated for butyramide is shown in figure 2.3.

The integer identifiers calculated for each atom correspond to an “on” bit in a  $2^{32}$  bit bit-set. Where the presence of a particular bit corresponds to the presence of a particular molecular feature [Rogers and Hahn, 2010]. As further iterations expand the radius of information for each atom and more features are identified, the hash function calculates the corresponding bits to set and these new identifiers are appended to the fingerprint set for the molecule.

For example, in figure 2.3, the initial fingerprint, “ECFP\_0” indicates that bit 734603939 should be set in the fingerprint for butyramide. This corresponds to the presence of a single methyl group in the molecule. After the second iteration this identifier is updated to 863188371 indicating the presence of a methyl carbon bonded to an ethyl carbon.

## Tanimoto Similarity

The Tanimoto coefficient (also referred to as the Jaccard index), given in equation 2.1, is a metric for quantifying the similarity of sets. It calculates the ratio of the intersection of two sets to their union, and is proportional to their similarity [Nikolova and Jaworska, 2004], that is, the ratio of the number of features which the sets share to the number of features which they do not share.

$$\tau(A, B) = \frac{N_{A\&B}}{N_{A|B}} = \frac{N_{A\&B}}{N_A + N_B - N_{A\&B}} \quad (2.1)$$

Where  $N_A$  is the number of features in set  $A$ ,  $N_B$  is the number of features in set  $B$ , and  $N_{A\&B}$  is the number of features both  $A$  and  $B$  share, that is, the intersection of  $A$  and  $B$ .

Conversely, the dissimilarity between two sets is quantified by their Tanimoto distance [Nikolova and Jaworska, 2004], given by

$$d(A, B) = 1 - \tau(A, B). \quad (2.2)$$

When applying this to ECFPs, the sets of identifiers shown in figure 2.3 are the sets  $A$  and  $B$  in equation 2.1. Where each fingerprint set corresponds to the “on” bits in a bit bit-string with up to  $2^{23}$  bits. Less bits are often used however, in this research 2048 bit ECFP\_4 fingerprints were used.

These fingerprint sets are converted to their bit-string representations and a simple method calculates the number of intersecting bits between them [RDKit, 2021b].

Furthermore, Tanimoto Similarity can be used to calculate the similarity between sets of molecules. This is usually done in two ways. First, the diversity of structures within a set is referred to as the internal diversity [Polykovskiy et al., 2020], and it is calculated as

$$IntDiv = 1 - \sqrt{\frac{1}{N_A^2} \sum_{m_1, m_2 \in A} \tau(m_1, m_2)}. \quad (2.3)$$

This measures the average dissimilarity, or diversity, between every pair of molecules in a set.

Second, the similarity between molecules in one set and those in separate set is referred to as the external similarity [Polykovskiy et al., 2020], and this is calculated with as

$$ExtSim = \frac{1}{N_A} \sum_{m_A \in A} \max_{m_B \in B} \tau(m_A, m_B). \quad (2.4)$$

This measures the average similarity between each molecule in set  $A$ , and it's nearest neighbour (or most similar molecule) in set  $B$ .

## 2.4 QSAR and Combinatorial Synthesis Approaches

While structural analogues can be expected to share similar **BA** and **selectivity** for **receptors**, other pharmacological properties that are significantly relevant for drug molecules are due to physical properties of drugs, properties which can be similar for significantly different structures.

For example, the water-octanol coefficient of a molecule is measured by dissolving the compound in mixture of 1-octanol and water and measuring the compound's relative dissolution between the two layers.

Water and 1-octanol are chosen because water is an analogue for the aqueous blood system and 1-octanol is similar in structure to the lipid membrane of human cells [Bruice, 2011].

For example, it has been shown that for a compound to be able to have effective distribution in the body, that is, to reach target cells and cross the cell's lipid barrier it requires a water-octanol partition coefficient within an effective window. If the coefficient is too low the compound will be unable to penetrate the cell membrane. If the coefficient is too large, the compound is unable to cross the aqueous blood phase to reach the target cell [Bruice, 2011].

Water is a polar molecule whereas 1-octanol is non-polar, therefore a molecule's partition coefficient is directly determined by its **polarity**. Therefore, features of molecular structures which determine **polarity** are relevant to whether or not a molecule would have desirable distribution properties.

Therefore, being able to use molecular structure to predict physical properties which are known to track biological activity would allow the design of compounds with desired activity profiles. Such relationships are referred to as **quantitative structure-activity relationships (QSARs)**.

These relationships are obtained by collecting sets of compounds which are known to have a certain biological activity and identifying the common structural features which could be the cause of the property.

For example, equation 2.5 is a **QSAR** that was obtained experimentally from a set of dihydrofolate reductase inhibitors obtained by substituting the 2,4-diaminopyrimidine **scaffold** shown in figure 2.4 [Bruice, 2011].

$$potency = 0.80\pi - 7.34\sigma - 8.14 \quad (2.5)$$

Where  $\sigma$  is the electron donation potential of the substituents and  $\pi$  is the **hydrophobicity** of the substituents. This relationship elucidates that for the **receptor** in question, **ligand** potency was found to be proportional to **hydrophobicity** and electron donation. Obtaining such relationships

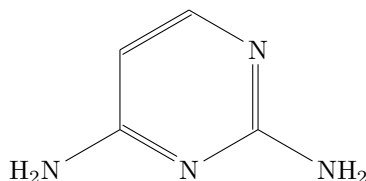


Figure 2.4: 2,4-diaminopyrimidine scaffold for which the QSAR given in equation 2.5 was obtained by measuring the potency of substituted variants of this scaffold as inhibitors for the dihydrofolate reductase enzyme [Bruice, 2011].

informs the drug design process because, for example in this instance, an optimal ligand can be designed by substituting the scaffold in figure 2.4 with the appropriate substituents.

In order to obtain the measurements to model such relationships libraries are created from molecular building blocks. All permutations of these building blocks are synthesized, and their activity is measured *in vitro*.

## 2.5 Evolutionary Approaches

Evolutionary computing (EC) is the application of EAs to the solution of multidimensional optimization problems with very large solution spaces.

EAs are based on the principle of natural selection where, starting with a population of randomly initialized solutions, a fitness function is used to evaluate how well each solution solves the problem at hand. Solutions have the potential to be improved by having their parameters crossed over with those of other solutions, or through random mutation. By linking the likelihood of a solution being selected for crossover to its fitness, and having the offspring compete with the current generation for a place in the population, the fitness of the population increases over successive generations [Eiben and Smith, 2015]. The algorithms are inspired by how the evolutionary process can be extended beyond natural selection to general purpose problem-solving algorithms, or metaheuristics. This metaphor of evolution as a problem-solving algorithm is shown in table 2.1.

In the context of this research, drug design can be treated as an optimization problem, where the molecule is the input and its properties are the output. Given a specified target range for the properties, the aim is to find the set of molecules which sufficiently satisfy these constraints, thus identifying new candidates for synthesis and testing.

EC has been applied to drug design and optimization in a wide variety of ways, including the generation of libraries of novel molecules with a sufficient degree of similarity to a set of reference structures. For example, the method was used to evolve libraries of compounds with a sufficient degree of similarity to a particular class of ligands, angiotensin-II antagonists, as well as candidates for ligands which were likely to bind to the active site of the stromelysin-1 protein [Sheridan et al., 2000]. Another approach started from a population of minimal molecules represented with a graph-based genome, thus not biasing the initial population to any particular region of chemical space, which grow through mutations which add, remove or exchange atoms, form or break rings, amongst other operations. The fitness function was a weighted sum of molecular properties calculated from the graph, and the fit of the graph to a predefined grid, which served as an estimate of the candidates' BA as a ligand for a target receptor [Glen and Payne, 1995]. This was used to design molecules with a minimum degree of similarity to ribose and molecules which could be expected to have an affinity for the bacterial enzyme dihydrofolate reductase.

Alternatively the method has been used to find optimal conformations for known molecules, for example optimal ligand conformations were evolved for a set of 100 known protein-ligand complexes, using fitness functions which directly calculate a genotype's BA [Jones et al., 1997].



Table 2.1: Evolution as a metaphor for problem-solving [Eiben and Smith, 2015].

Evolution		Problem-Solving
Environment	↔	Problem
Individual	↔	Candidate Solution
Fitness	↔	Quality

Here the chromosomes encoded a known **ligand**'s torsion angles and hydrogen donors and acceptors, where the fitness function calculated a least-squares fit of the hydrogen donors and acceptors represented by the genotype and those of the **receptor**, as **hydrogen bonding** is one factor which determines the location and orientation of a **ligand** with respect to the **receptor**. The internal energy of the **ligand**, calculated from its torsion angles, and the **ligand-protein** interaction, calculated from the **hydrogen bonding** information, is used to measure the candidate's docking fitness, where a lower total internal energy indicates a higher **BA** for the **receptor**. Another approach [Supady et al., 2015] generated a population of random conformers for a given **SMILES** string, represented by a vector of torsion angles, where the fitness function calculated each individual's conformational energy, estimated with Density Functional Theory [Atkins and de Paula, 2010], and modulated relative to the other individuals in the population. This method was used to generate low energy conformers of a set of amino acid dipeptides and a larger mycophenolic acid molecule.

The method has also been used to optimize **physico-chemical** properties relevant to specific applications, such as the evolution of optimal molecular switches for use in photochemistry, where a given backbone was modified in order to find the composition of substituents producing a molecule with an optimal excitation energy [Ole Carstensen et al., 2011], and molecular dipole moment [Rupakheti et al., 2015] where an evolutionary landscape defined by 40 dimensional Moreau-Broto autocorrelation descriptors. AM1 quantum mechanical calculations were used to estimate each genotype's dipole moment and a genetic algorithm was used to generate a population of diverse compounds with dipole moments above  $5.5D$ . Diversity was measured as the euclidean distance in chemical descriptor space.

Another example, ChemGE [Yoshikawa et al., 2018], employed population-based grammatical evolution to an initial population of **SMILES** strings corresponding to known **ligands**, and a fitness function which calculated an individual's **DS** with thymidine kinase using rDock [Ruiz-Carmona et al., 2014]. The method produced a diverse set of 189 **ligands** with better **DSs** than the best known **ligand** in the DUD-E database [Mysinger et al., 2012].

Furthermore, a comparison between a graph-based genetic algorithm, **monte-carlo tree search** (**MCTS**), and **VAEs** on the problem of logP optimization [Jensen, 2019] demonstrated that the graph-based genetic algorithm generated results that were as good or better than those produced from the **MCTS** or **VAE** methods while being orders of magnitude more computationally efficient.

## 2.6 Reinforcement Learning Approaches

**RL** is arguably the most successful learning paradigm, distinct from supervised and unsupervised learning, and which most closely mirrors how humans and animals learn. The field draws on multiple areas of psychology, specifically classical Pavlovian conditioning, temporal-difference learning, the reward prediction error hypothesis, and the neurological role of dopamine [Richard S. Sutton and Andrew G. Barto, 2018]. It is a field which tries to understand the general processes underlying learning and intelligence.

The various **RL** algorithms have been very successful in their application to unknown environments, where the reward function is unknown to the agent and must be learned through exploration. This makes it a flexible method, applicable to almost any learning problem, and a viable path to obtaining generally intelligent systems, capable of learning at least a human level of skill on a wide

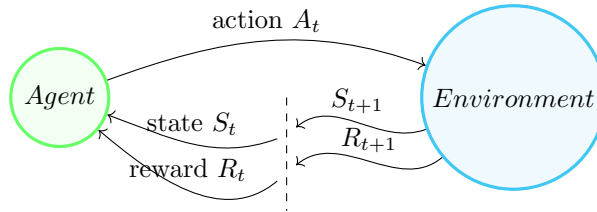


Figure 2.5: The interaction of an agent with its environment in a Markov decision process (MDP).

range of tasks.

If a problem can be described as a Markov decision process (MDP), shown in figure 2.5, that is, one where the actor or agent can observe the state of the environment, and based on the information in the current state, take an action in accordance with some goal, then a solution to that problem is considered an RL method [Richard S. Sutton and Andrew G. Barto, 2018].

There are various aspects to consider when designing an RL algorithm, and the different approaches to these lead to variations of the fundamental method. These considerations include whether the most appropriate approach would be model-based or model-free, value-based or policy-based, on-policy or off-policy and how to bootstrap learning. A discussion of these concepts is beyond the scope of this thesis [Richard S. Sutton and Andrew G. Barto, 2018], but it is important to note that popular RL algorithms are designed to deal with classes of problems that are defined by these considerations.

## Value-Based Reinforcement Learning

Both value-based and policy-based methods attempt to learn a policy which maximizes the return over an episode  $T$

$$R_T = \mathbb{E} \left[ \sum_{t=1}^T r_t \right] \quad (2.6)$$

received by the agent when deployed in the environment. Value-based RL methods attempt to first learn to predict the state-action value function

$$Q(s_t, a_t) = \mathbb{E}_t [R_t | s_t, a_t]. \quad (2.7)$$

Which returns the expected cumulative reward for any given state-action pair, and from this they extract a policy, where the policy is to choose the action in each state with the greatest value.

The dominant value-based RL algorithm is Q-learning which implements value iteration using the Bellman function

$$y_t = r_t + \gamma Q^*(s_{t+1}, a_{t+1}) \quad (2.8)$$

to predict the target state-action values for a given transition. The agent obtains transitions through  $\epsilon$ -greedy exploration of the environment.

In its simplest form the algorithm is implemented in tabular form with the agent iteratively updating its value estimates for each state as it obtains new information from the environment. Recently the method has been implemented with ANNs which are able to learn mappings for very large state spaces which would require prohibitive storage space to implement in tabular form [Mnih et al., 2013, Mnih et al., 2015]. These neural implementations, referred to as deep Q-learning, leverage a replay memory of past exploration trajectories and learn to approximate the value-function with backpropagation through gradient-descent [Rumelhart et al., 1986].

The loss function for gradient descent is therefore the difference between the targets predicted by equation 2.8 and those predicted by the ANNs  $Q(s_t, a_t)$

$$L = y_t - Q_t. \quad (2.9)$$

Value-based RL has successfully been applied to the generation of molecular graphs optimized for solubility, QED and SA [Zhou et al., 2019].

## Policy-Based Reinforcement Learning

Policy-based methods attempt to learn an optimal policy directly, without first learning a value-function from which the policy is extracted. That is, while some policy-based methods may integrate learning of an approximation of the state-action value function, the policy is a probability distribution of the action space for a given state, such that sampling from that probability distribution will maximize the return received (defined by equation 2.6) over an episode  $T$ .

As with deep Q-learning, recent policy-based methods benefit from a neural implementation and are thus referred to as policy-gradient methods. There are a number of popular policy-gradient methods, such as REINFORCE [Richard S. Sutton and Andrew G. Barto, 2018], which learns only a parametrized probability distribution over the action space without any dependence on the state-action value function.

While a discussion of the policy-gradient methods is beyond the scope of this thesis, they all implement a variation of the policy gradient theorem [Richard S. Sutton and Andrew G. Barto, 2018] to define the parameter update for the policy network

$$\theta \leftarrow \theta + \alpha G_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \quad (2.10)$$

Actor-critic methods [Konda and Tsitsiklis, 2003] such as deep deterministic policy gradients [Lillicrap et al., 2015] and proximal policy optimization [Schulman et al., 2017] integrate value-based and policy-based learning by learning the state-action value function (the critic) simultaneously and using this to bootstrap the policy (the actor).

Keeping the definition of an MDP in mind, and applying it to drug design, the environment is chemical space, the space of physically realizable molecules, or subsets of thereof. Any given molecule in chemical space is one state in the environment, and actions are those molecules obtained from modifying the present state. Stated this way, RL methods are readily applicable to the problem of drug design.

## Generative Adversarial Networks Incorporating Reinforcement Learning

There have been numerous applications of RL to drug design which have incorporated GANs to sequentially map from a continuous vector to a discrete representation of a molecule, with the output being recurrently combined with the input vector to generate the next step of the output.

Here a separate ANN referred to as the discriminator alternates with the generator during training where the two networks compete in a min-max game defined by equation 2.11 [Goodfellow et al., 2014].

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_{synthetic}(z)} [\log(1 - D(G(z)))] \quad (2.11)$$

The two networks alternately perform gradient descent via backpropagation to minimize their respective errors where the result is that the output of the generator  $G(z)$  eventually resembles the training data  $x$  as the generator network has learned the distribution of features which characterize elements of the training set.

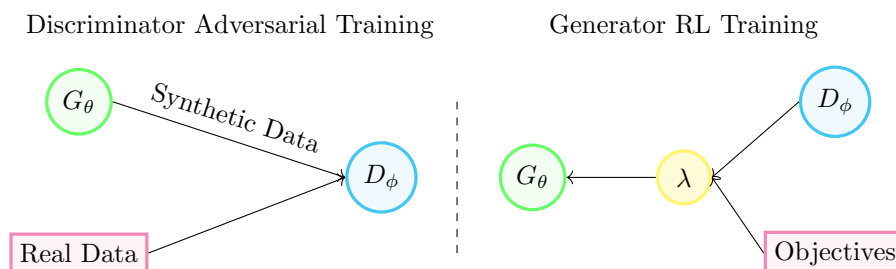


Figure 2.6: Adversarial training in combination with RL [Guimaraes et al., 2017]. Left: shows the standard adversarial training arrangement. Right: shows its integration with RL.

This process has been integrated with RL so that the reward to the generator network is a weighted sum of both the discriminator error and physico-chemical properties such as solubility, SA, and QED among others [Guimaraes et al., 2017] [Sanchez-Lengeling et al., 2017].

This integration of adversarial training (GANs) with RL is shown in figure 2.6.

A key complication when applying adversarial training to generate discrete structures however, is that the training process becomes non-differentiable. That is, as shown in equation 2.12, one can only assign a discriminator reward to the final action which completed the sequence.

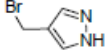
$$Q(s = Y_{T-1}, a = y_T) = R(Y_T) \quad (2.12)$$

This complication is solved by using N-time MCTS with  $G_\theta$  as the rollout policy to estimate the intermediate state-action values  $Q(s = Y_{t_i}, a_{y_{t_i}})$  [Yu et al., 2016].

This method was used to generate SMILES strings of molecules which, in terms of their constitution, resembled those of a set of reference molecules and simultaneously had improved solubility ( $\log P$ ), SA and QED [Guimaraes et al., 2017]. Here  $G_\theta$  was composed of recurrent neural network (RNN) layers with long short-term memory (LSTM) cells which mapped from a continuous input vector to a SMILES sequence.  $D_\theta$  was composed of one-dimensional convolutional layers and designed for text classification.

The result being a network which, given any continuous vector input, would recurrently construct a novel SMILES string similar in constitution to examples from the training set, and possessing physico-chemical properties within a desired range.

In an extension to this approach, adversarial training in combination with RL was used to train an agent to generate an optimized set of molecules, except here the set was optimised for melting point, and their use as photovoltaics [Sanchez-Lengeling et al., 2017]. For melting point optimization an auxiliary ANN trained on a set of experimentally determined molecular melting points provided the reward signal. For photovoltaic potential another auxiliary network was trained on molecules for which experimental data was available for their photoelectric conversion efficiency, and this network was used to provide the reward signal to the generator during adversarial training on a separate dataset of organic photovoltaics. The trained network was able to generate a set of novel molecules similar in constitution to known organic photovoltaics but with a higher average expected power conversion efficiency as estimated with quantum chemistry methods. Here the generator also used recurrent neurons to sequentially construct SMILES representations of molecules which were input to the network as hot-encoded binary matrices; this is a representation where every character in the SMILES string is represented by a fixed-length bit-vector, with a single bit being set which corresponds to the character. An example of a hot-encoded SMILES matrix is shown in figure 2.7.

Graph 

SMILES BrCc1c[nH]nc1

One-hot encoding

	Br	C	c	1	c	[	n	H	]	n	c	1
Br	1	0	0	0	0	0	0	0	0	0	0	0
C	0	1	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	1	0	0	0	0
c	0	0	1	0	1	0	0	0	0	0	1	0
n	0	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	1	0	0	0	0	0	0	0	1
[	0	0	0	0	0	1	0	0	0	0	0	0
]	0	0	0	0	0	0	0	1	0	0	0	0

Figure 2.7: An example of hot-encoded SMILES matrices [Blaschke et al., 2018]. The rows correspond to characters in the SMILES string and the columns correspond to the set of possible SMILES characters, with the number of possible characters determining the length of the bit-vector. Such representations are commonly used as inputs to recurrent ANNs.

## 2.7 Autoencoding Approaches

Autoencoders (AEs) are a class of ANNs which copy their input  $x$  to their output  $r$  by passing it through a lower dimensional hidden layer (latent code)  $h$ . In order to do so, the ANN has to learn one mapping  $h = f_{\theta}(x)$  from the input to the lower dimensional hidden layer (the encoder), and a second mapping  $r = g_{\phi}(h)$  which reconstructs a sample from a given lower dimensional code (the decoder) [Goodfellow et al., 2016]. This general structure is shown in figure 2.8. The network learns these two mappings simultaneously during training by learning the features that define the training data. Once trained, the generator can be given arbitrary latent codes, from which it will generate new samples not present in the training data.

The use of a lower dimensional hidden layer prevents the ANNs from perfectly copying the input to the output. This constrains the network to identify the features which will maximize the accuracy of the reconstruction over less impactful features. These statistically salient features are those which define the training data.

The network is trained by attempting to minimize the reconstruction error over a set of training samples, and how this reconstruction error is defined will depend on how the data is represented.

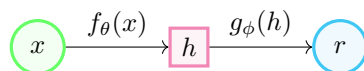


Figure 2.8: AE structure. The encoder  $f_{\theta}(x)$  maps the input  $x$  to an internal representation  $h$  and the decoder  $g_{\phi}(h)$  maps the encoding to a reconstruction of the input  $r$ .

### Variational Autoencoders

However, when working with deterministic mappings like this, many arbitrary latent codes will yield invalid samples when decoded.

This problem can be ameliorated by taking a Bayesian approach, so that the encoder learns  $p_{\theta}(h|x)$  instead of a deterministic mapping  $h = f(x)$ . That is, given a point  $x$ , the encoder generates

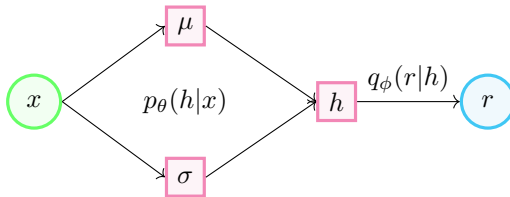


Figure 2.9: VAE structure. The encoder  $p_{\theta}(h|x)$  maps the input  $x$  to a sample  $h$  drawn from the prior distribution  $p(h)$ , and the decoder maps the sample  $h$  to a posterior probability over the output domain  $q_{\phi}(r|h)$ .

a distribution over the possible values of the latent code  $h$  from which  $x$  was likely sampled. The distribution here is a multivariate Gaussian  $\mathcal{N}(\mu, \sigma^2)$ , where  $\mu$  and  $\sigma^2$  are vectors. The decoder similarly learns  $p_{\phi}(r|h)$  instead of  $r = g(h)$ . That is, given a latent code  $h$ , the decoder generates a distribution over the possible values of an instance  $x$  [Kingma and Welling, 2013].

This can be done by having the latent code be sampled from a vector of distributions instead of deterministic elements. This is done by splitting the hidden layer into two hidden layers:  $\mu$  and  $\log \sigma^2$  which correspond to vectors of the means and the log of the variance of these distributions. From these two vectors a latent code is sampled. This structure, referred to as a **variational autoencoder** (VAE), is shown in figure 2.9.

The mapping from an instance  $x$  to a multivariate distribution defined by the vectors  $\mu$  and  $\log \sigma^2$  is then learned by introducing the Kullback-Liebler divergence into the loss function alongside the reconstruction error [Kingma and Welling, 2013]:

$$L = KL + R(x, g_{\phi}(f_{\theta}(x))) \quad (2.13)$$

$$KL = \sum_{i=1}^n 1 + \log \sigma_i^2 - \mu_i^2 - e^{\log \sigma_i^2}. \quad (2.14)$$

where  $R$  is the reconstruction error. The **Kullback-Leibler** (KL) term corresponds to the sum over the elements of hidden vectors  $\mu$  and  $\log \sigma^2$ , where each element  $i$  corresponds to the log variance and mean of the distribution for element  $i$  of the latent code. The KL Loss is the sum of the divergences between multivariate distribution defined by  $p_{\theta}(z|x)$  and a normal distribution  $N(\mu_i, \sigma_i^2)$ . Minimizing the KL loss therefore means optimizing the weights of the encoder such that the probability distribution parameters  $\mu$  and  $\sigma^2$  resemble those of a normal distribution. Sampling new latent codes from the distribution represented by vectors  $\mu$  and  $\log \sigma^2$  improves the validity of samples generated by the decoder.

The result of these modifications is that the encoder learns to calculate the probability  $p(h|x)$ , that is, it models the probability distribution over all possible values for the elements of the latent vector  $h$  given  $x$ . The latent code  $h$ , the elements of which are sampled from the distributions defined by the hidden vectors  $\mu$  and  $\log \sigma^2$ , is then taken to be the prior probability for the generator. The generator then learns to calculate the posterior probability  $q(r|h)$ , that is, the probability of the possible constituents of a reconstructed sample given a prior probability  $p(h)$ . When working with images this would be a posterior distribution over the RGB values for each pixel. The KL term regularizes the prior probability distributions  $p(h)_i$  defined by  $\mu_i$  and  $\sigma_i^2$  by forcing them to be normally distributed. Therefore, sampling new latent codes  $h$  as a vector of normally distributed elements is more likely to lead to valid reconstructions.

VAEs have been applied to the generation of novel SMILES strings using networks composed of one-dimensional convolutional layers of recurrent neurons [Harel and Radinsky, 2018]. Here one-dimensional recurrent convolutional layers were used to map from SMILES strings to a continuous vector encoding. Latent codes were decoded to SMILES strings by mapping from the continuous vector to a probability distribution over the available characters. This was done iteratively for

each character in the sequence until the terminal token was chosen. Given the generator produces a probability distribution, from which the next token is sampled, it is possible for the same latent code to generate multiple strings. Here the authors mimicked how chemists will often use a known reference compound as a prototype for drug design by encoding a prototype as a latent vector and sampling the local neighbourhood by adding noise to the prototype code. Due to working with SMILES strings however, and using an unregularized latent space, a large proportion of their generated samples were invalid.

This method was expanded upon by using Bayesian optimization to find latent codes which decode to molecules with optimal properties [Gómez-Bombarelli et al., 2018]. Here the authors also iteratively decoded a SMILES string from a continuous latent code by generating a probability distribution over the available strings and sampling a character from the distribution. However, here a separate ANN was trained to predict pharmacological properties from latent codes jointly during reconstruction training. Novel latent codes were then sampled by encoding a reference molecule and moving in the direction in latent space, predicted by Bayesian optimization to be most likely to improve the property of interest.

## Conditional Generation

Conditional generation decodes a latent code given a particular class. Vanilla VAEs model  $p(z|x)$ , they make no consideration for the class of  $x$ . Similarly, the decoder models  $p(x|z)$  with no knowledge of class. Letting class be  $c$ , a conditional variational autoencoder (CVAE) models  $p(z|x, c)$  and  $q(x|z, c)$ . This is called *conditioning* the encoder and decoder on the variable  $c$ . The distribution for latent codes then becomes  $p(z|c)$ . That is, for any given  $c$  there is a multivariate distribution  $p(z)$ . Therefore, one can sample a code from the latent layer of the encoder and present it to the decoder with the desired class concatenated in order to generate a novel instance of the desired class. The same latent code can be concatenated with different labels to generate instances from different classes. Or multiple latent codes can be concatenated with the same class to generate multiple molecules possessing a desired property.

Conditional generation was used to generate structural analogues of known ligands by using measured activity for a target receptor as the label. The authors collected a set of JAK2 and JAK3 inhibitors from ChEMBL and used experimentally measured activity for these two targets as the conditioning class. Then by setting the generation condition to high activity against JAK3 and low activity against JAK2 they generated 300,000 candidates. After passing these candidates through a series of filters and simulations they reduced this to 100 promising candidates. A group of experts then selected their preferred candidate from this set to be tested *in vitro* [Polykovskiy et al., 2018].

In other work adversarial training was used to bias the generation towards regions of chemical space containing molecules similar to a set of known anti-cancer drugs [Kadurin et al., 2017]. The authors constructed a conditional adversarial autoencoder (AAE) by concatenating drug concentration to the input and output fingerprint vectors, as well as concatenating tumour growth inhibition (GI) percentage to the latent code  $z$ . They trained the model on samples for which these measurements were available as measured from bioassays of the MCF-7 tumour cell line. Once trained 640 latent vectors were sampled from the encoder’s prior distribution along with 640 GI values. These were presented to the decoder to generate 640 molecular fingerprints including their expected concentrations. During reconstruction training adversarial training was used to regularize the latent layer through discrimination with a normal distribution. The generated fingerprints were filtered for those with low drug concentration and the resulting fingerprints were screened against a library of 72 million compounds obtained from PubChem. From this screening 69 hits were obtained.

## 2.8 Discussion

As mentioned in section 2.2 the drug design problem is one of finding a solution to a proposition, where here the proposition can be defined by a set of functions, modelled by artificial neural networks, which map molecular structure to physico-chemical properties and biological activity, where these models are trained through supervised learning from experimental measurements. The goal then is to design a molecule or set of molecules which constitute a Pareto front with respect to this set of constraints.

Bearing in mind that the true potential of intelligent systems applied to this context lies in their learning ground-truth relationships obtained not from models and simulations developed by humans, but from experiences obtained from exploration in the real-world robotic laboratories. This section presents the argument that deep RL is superior in this context compared to EAs, AE approaches and GANs which include RL.

Secondly, as a foundation for the experiments described in chapter 3 it describes a method for improving the performance of deep RL applied to automatic drug design by incorporating a strategy for mitigating the sparse rewards problem.

As already discussed in section 2.2 linear search through all known small molecules is not a viable approach, and this method is not meaningfully improved by simply being more selective about the chosen subsets.

One could try a random-walk approach which constructs molecules from random combinations of molecular building blocks and returns only those which pass all defined filters. For example, QED scores a molecule according to the presence and absence of certain functional groups, so the groups which are beneficial to this score could be set as essential building blocks for the random walk algorithm. However, this method has no guarantee of optimal convergence.

One needs to find the regions of chemical space which are likely to pass the given set of filters. Random walk could be improved with EAs which encode these molecular building blocks as genes and through mutation and selection the algorithms are able to find those genes which are likely to satisfy the given filters, but this method is also not guaranteed to do so efficiently or optimally.

AEs attempt to model the features of a given set of reference molecules, such as those with properties within a useful range and through recurrent decoding, generate novel structures with similar properties.

It is possible to use AEs to learn a mapping from discrete molecular representations to continuous vectors, and train an auxiliary network to predict multiple physico-chemical properties from the latent layer, thus allowing one to use Bayesian optimisation to obtain latent codes which optimally satisfy a multivariate reward function, codes which can then be decoded back to discrete molecular descriptors. However, due to the semi-supervised nature of this training model, the results will always be biased to the same region of chemical space as the training data. It is also difficult to apply this model to the generation of molecules optimized as ligands for specific targets, as for such applications there are significantly fewer examples of positive examples for the model to learn from. Without sufficient positive training samples, autoencoding methods will struggle to produce useful results. While there have been, as discussed in section 2.7, methods of using conditional and AAEs to solve this, the results of these methods do not seem to warrant the significantly increased complexity.

There are six motivating factors for the preference towards deep RL over GANs incorporating RL. The first is the difficulties inherent in training GANs due to the nature of their training process. The second is the sample-dependent nature of training GANs. The third is the theoretically superior performance of deep RL agents over GANs, and the fourth is the experimentally demonstrated superior performance of deep RL algorithms over GANs on drug optimization tasks. The fifth is that a reusable policy, as is obtained from deep RL training, is more appropriate for the intended applications of an autonomous self-improving robotic laboratory. Six is the increased flexibility when working with deep RL methods as opposed to GAN-RL methods. These six concerns are



unpacked below.

GAN optimization is inherently unstable due to the nature of the training problem, as it is an attempt to locate a saddle point between the performance of the generator and discriminator, leading to difficulties such as the perfect discriminator problem, mode collapse or vanishing gradients [Elton et al., 2019]. Thus, simpler training methods are preferable, especially if they can be made to be more performant.

GANs, like AEs, are supervised learning algorithms. That means that in order to train them, they require a large source of labelled training samples from which to learn. In the context of drug design where the goal is to generate novel compounds for which there are no available reference sets, supervised learning algorithms are not appropriate. If an autonomous agent is attempting to learn to generate ligands for a target receptor, there likely will not be a sufficiently large set of positive samples to train the discriminator. Deep RL on the other hand is capable of learning entirely unsupervised, improving its performance using only the feedback it receives from the environment through its own exploration, thus requiring no positive examples in order to learn to generate a ligand of the required class. Secondly in cases where there are limited reference sets, supervised learning algorithms attempt to generalize from the samples that they have seen about those which they have not. GANs for example will attempt to generate novel compounds which are structurally analogous to a reference set through the use of a discriminator. However, that reference set constitutes only a tiny fraction of the estimated  $10^{60}$  drug-like small molecules. Those which have a constitution and configuration which is known to be appropriate for the intended use. However, there may be other combinations of constitution, configuration and conformation not contained within the reference set which are also appropriate and potentially superior for the goal. Therefore, attempting to generate analogues of known leads is unlikely to discover candidates which have the desired properties but dissimilar structural features to the reference set. Limited training data is therefore likely to lead to pure deep RL approaches being the preferred method in future applications of ML to drug design [Elton et al., 2019].

Regarding experimentally verified performance, the MolDQN framework [Zhou et al., 2019] and the graph-convolutional policy network (GCPN) [You et al., 2018] have both been shown, when optimizing properties such as LogP and QED, to have superior performance to GAN based RL methods ORGAN [Guimaraes et al., 2017] and ORGANIC [Sanchez-Lengeling et al., 2017], as well as the leading AE method: junction tree variational autoencoder (JTVAE) [Jin et al., 2018, Elton et al., 2019].

Between the two deep RL methods, the GCPN employed an adversarial training method which was later outperformed by MolDQN using only Tanimoto Similarity.

The added benefit of the deep RL approach is that it is more amenable to targeted property optimization, as a trained agent can be given any lead candidate whereupon the agent will apply its policy in order to modify the given molecule such that the physico-chemical properties are modified according to the agent’s training goals.

Another interesting result from the MolDQN experiments [Zhou et al., 2019] was that the use of Tanimoto Similarity proved to be sufficient for the constrained generation of molecules and in fact yielded better results than the GCPN which employed adversarial training to constrain optimization by maintaining similarity [You et al., 2018]. The benefit of Tanimoto Similarity is that it can be incorporated into the reward signal to both maintain similarity to a given starting molecule in the case of lead optimization, or by using equation 2.4, similarity to a reference set can be maintained without the difficulties of adversarial training such as mode collapse and the perfect discriminator problem.

Under the RL method the agent learns which actions during the construction of a molecule will lead to an optimal reward. This yields a policy which can be used to both optimize any given molecule, or to generate novel molecules. If an agent learns a policy which will maximize a reward composed of physico-chemical properties and DS, this can be used to generate an optimal candidate by starting from an empty molecule and greedily choosing the modification expected

to yield maximum reward, that is, the action with the highest  $Q$ -value. An RL agent can also generate a set of viable candidates by choosing actions greedily with probability  $\epsilon$ , without needing to train a generator through adversarial training. Alternatively, random molecules can be chosen from significantly different regions of chemical space: from known sets, or generated by an AE, and the agent’s learned optimization policy can be applied to bring them into the region of optimal physico-chemical properties and BA.

The final consideration references the context where this research is situated: towards the goal of autonomous robotic laboratories. RL is more appropriate than the other methods discussed in this chapter for this application as it easily extended from simulation to a real-world laboratory where the robotic scientists generate experimental metadata and measurements through automated experiments from which they improve their predictions for subsequent experimental parameters pursuant to their goal.

Referring back to the research questions posed in section 1.2, the related work reviewed in this chapter indicates that RL is indeed a viable method of training an intelligent agent to autonomously optimize molecular scaffolds for a specific target receptor through exploration in simulation. The RL method is particularly suited for the hit-to-lead phase of drug discovery and design [Sumita et al., 2018, You et al., 2018, Zhou et al., 2019, Khemchandani et al., 2020] compared to alternative methods such as EAs, AEs or GANs, which are better suited for the hit-generation phase. Furthermore, the RL method is particularly appropriate considering the context of autonomous scientific laboratories [Elton et al., 2019]. The methodological details of how deep Q-learning, specifically the double deep Q-networks (DDQN) algorithm with reward shaping, was implemented for this task of ligand generation are given in the following chapter, and the performance of the algorithm on this task is reported in chapter 4.

## Chapter 3

# Methods

The methods used in this work build on similar applications of RL to automated drug design, particularly the use of Q-learning [Zhou et al., 2019], which was explored in its application to the generation of molecules with optimized physico-chemical properties such as  $\log P$ , QED and SA; which were measured with the RDKit cheminformatics package. The work thus demonstrated the algorithm’s ability to learn these rules, and to optimize molecules according to constraints imposed on these properties while retaining similarity to the starting structure.

However, in order to develop a ligand for a particular target receptor one needs to measure a given molecule’s BA and specificity for that receptor. This can only be accurately measured experimentally, but it can be estimated through simulations which are becoming increasingly more reliable. Therefore, this work seeks to incorporate simulations of molecular docking for a receptor identified as a target for a specific pathology into the reward function.

In summary, the experiments focus on the application of deep Q-learning to train an agent through docking simulations to optimize a ligand’s BA for a protein target, using DS as an estimate.

This learning framework was applied to four case studies. The first was QED optimization (section 4.1), where the agent constructs ligands from scratch, and attempts to find a path through chemical space to a ligand with maximal QED. The second was DS optimization when learning from sparse rewards (section 4.2.1), where the agent attempted to generate ligands with maximal DS for a target receptor when only the terminal state in a rollout episode was rewarded. This was done to minimize total run-time as docking every intermediate state made the run-time unreasonably long. The third test applied reward shaping to ameliorate the impact of sparse rewards and facilitate learning under these conditions (section 4.2.2). The final test applied the algorithm composed of shaped sparse DS rewards beginning from three different chemotypes: naphthyridine, imidazopyridazine, and aminopyradine to assess the impact of starting from a region of chemical space known to contain structures with high DS for a target (section 4.2.3).

The following sections break down the components of the learning framework: the agent, environment, docking simulator, reward function and learning algorithm, and how they were applied in this context.

The specific target receptor, against which the agent was trained, was the adenosine triphosphate (ATP) binding site of the plasmodium falciparum phosphatidylinositol 4-kinase (PfPI4K) enzyme, shown in figure 3.1, an active target for which high-affinity ligands are still being sought. This is the site where ATP would bind and activate PfPI4K through conformational change in order to drive its biological processes, thus being hydrolyzed to adenosine diphosphate (ADP) in the process.

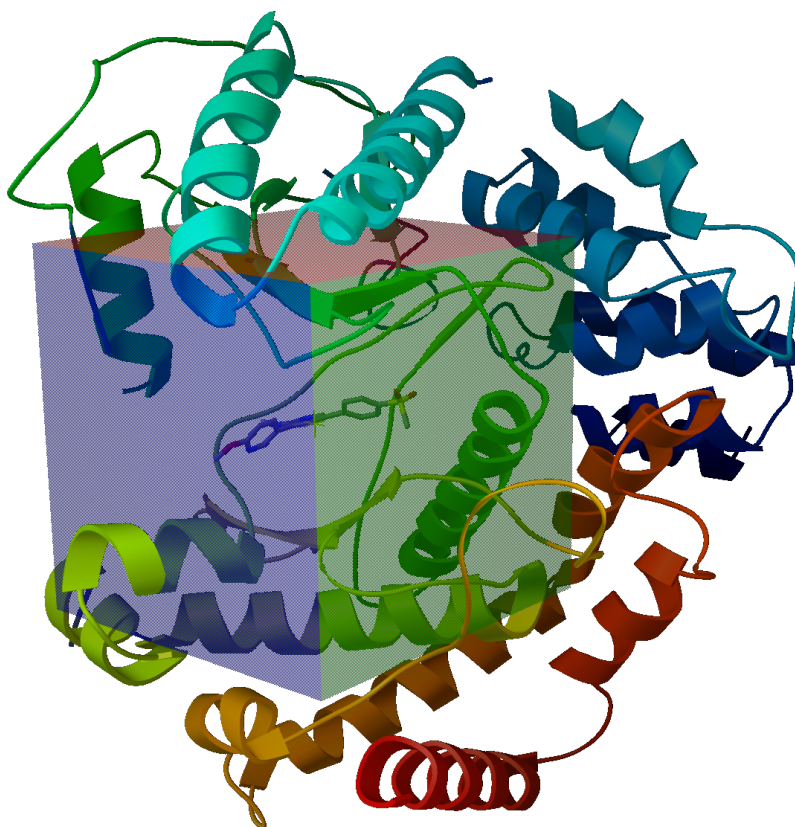


Figure 3.1: Ligand-protein training environment, showing the PfPI4K target enzyme [Fienberg et al., 2020] with a ligand docked at its receptor site. The receptor site is encapsulated within the docking box. All docking calculations are performed within the bounds of the docking box.

### 3.1 Agent

Under the RL model an agent explores an environment generating experiences from which it learns. A schematic of the interaction between the agent and environment is shown in figure 3.2. The agent here is implemented as a pair of fully connected deep ANNs, and a memory buffer, where the input to the network is the concatenation of three vectors.

The first is the 2048 bit ECFP\_4 molecular fingerprint of the current state of the graph, with each bit in the fingerprint corresponding to the presence of a certain functional group. Second is the fingerprint of each state that is accessible from the current state. These are concatenated to the current state in turn and for each, the network calculates the state-action value according to its Q-network. This approach is taken at each step of the graph design process as the number and nature of accessible states changes with the current state of the graph, so that it is impossible to define a constant action space. The action-value distribution over the states accessible from the current state is thus constructed by evaluating each accessible state in sequence. The final input parameter is the number of steps remaining in the design episode, such that each accessible state is considered relative to the number of steps the agent has left in which to modify the graph.

This input layer is followed by a number of one-dimensional convolutional layers for extracting features in the molecular fingerprints relevant to identifying patterns of functional groups in the molecules, which are subsequently followed a series of linear layers with rectified linear unit activations, with the number of neurons halving at each layer until the sigmoidal output layer, which

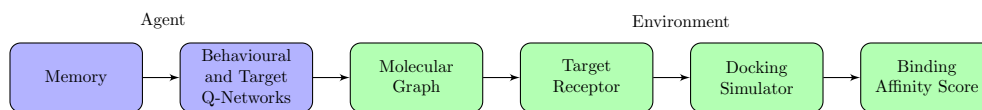


Figure 3.2: Algorithm summary showing the interaction of the agent and the environment. As the agent constructs molecular graphs they are evaluated through docking simulations with the target receptor, from which the agent receives the *DS* as a reward. These rewards are combined with the molecular graphs and stored in the agents' memory buffer to use for training.

contains a single neuron corresponding to the action-value of the state-action concatenation. This structure is the same for both the behavioural policy network ( $Q$ ) and the target network ( $Q^-$ ).

## 3.2 Environment

The training environment, shown in figures 3.1 and 3.2, is a combination of both the ligand which the agent is designing and the target protein, where their interaction is simulated with the Autodock-GPU docking package. During exploration the agent has a maximum number of steps in which to construct a candidate ligand. Once the design period is over docking simulations determine if it is possible for the given ligand to form a stable complex with the target protein, and if so, what is the change in the free energy of the ligand-protein complex as a result of binding. This change in binding free-energy determines the reward which the agent receives at the end of each exploration episode.

The agent designs molecules as two-dimensional molecular graphs using the `.mol` format, however it perceives these graphs as molecular fingerprints. At each step the agent is presented with the set of possible modifications it can make to the current graph, perceiving each as a molecular fingerprint and choosing the modification which it perceives according to its policy as leading to a molecule with the highest *DS* for the receptor site of the target in the remaining number of modification steps. After the modification episode the agent presents the ligand which it has designed to the target for docking.

### Simulating Molecular Docking

Molecular docking was simulated with the Autodock-GPU package. Given a molecular graph and the crystal structure of a target protein, docking proceeded as follows: the ligand's molecular graph was converted from its two-dimensional graph `.mol` format to its three-dimensional representation, including hydrogens, partial charges, and atomic coordinates using the `.pdbqt` format.

The crystal structure of the target protein was obtained from the protein data bank in `.pdb` format. This was also converted to its `.pdbqt` format. A three-dimensional docking grid was then prepared which encapsulated the receptor site of the target protein. The preparation of the target protein and docking grid was done once prior to training whereas the conversion of two-dimensional ligand graphs to their three-dimensional representations was performed after each molecular modification episode prior to docking. Given the `.pdbqt` ligand and protein files and docking grid, Autodock used its conformational search algorithm to explore the conformational states of the given ligand, evaluating the ligand-protein interaction for each conformation. This conformational search is implemented as a Lamarckian genetic algorithm [Morris et al., 1639]. The conformation with the greatest corresponding release of binding-free energy was then saved to a coordinate file. Autodock uses a semi-empirical scoring function (differentiated from knowledge based and physics based) which is a weighted sum of atomic interactions, tuned to structural data. Steric, hydrophobic, and hydrogen bonding interactions between atoms in the ligand in its current conformation, and atoms of the receptor within the docking grid are calculated. The weights of

these terms were computed from a non-linear fit of the scoring function to structural data [Trott and Olson, 2010].

The free energy  $\Delta G$  of a binding pose is given by

$$\Delta G = \Delta H_{vdW} + \Delta H_{hbond} + \Delta H_{elec} + \Delta G_{desolv} + \Delta S_{tor}. \quad (3.1)$$

Where  $\Delta H_{vdW}$ ,  $\Delta H_{hbond}$ ,  $\Delta H_{elec}$  are the enthalpy changes due to Van Der Walls interactions, hydrogen bonding and electrostatic interactions respectively,  $\Delta G_{desolv}$  is the Gibbs free energy change due to desolvation and  $\Delta S_{tor}$  is the change in ligand entropy due to the loss of rotatable degrees of freedom in the ligand as a result of binding.

These energetic terms are approximated semi-empirically as follows:

$$\begin{aligned} \Delta H_{vdW} &= W_{vdW} \sum_{i,j} \left( \frac{A_{ij}}{s(r_{ij})^{12}} - \frac{B_{ij}}{s(r_{ij})^6} \right) \\ \Delta H_{hbond} &= W_{hbond} \sum_{i,j} E(t) \left( \frac{C_{ij}}{s(r_{ij})^{12}} - \frac{D_{ij}}{s(r_{ij})^{10}} \right) \\ \Delta H_{elec} &= W_{elec} \sum_{i,j} \left( \frac{q_i q_j}{\epsilon(r_{ij}) r_{ij}} \right) \\ \Delta G_{desolv} &= W_{desolv} \sum_{i,j} (S_i V_j + S_j V_i) e^{-r_{ij}^2 / 2\sigma^2} \\ \Delta S_{tor} &= W_{tor} N_{tor} \end{aligned} \quad (3.2)$$

Where the sums  $\sum_{i,j}$  are over all intermolecular pairs of ligand-receptor atoms within the docking box.  $A_{ij}$ ,  $B_{ij}$  are constants which depend on the modified Lennard-Jones potentials [Atkins and de Paula, 2010] between atoms  $i$  and  $j$ , and  $C_{ij}$ ,  $D_{ij}$  are constants which depend on the hydrogen bonding potentials between  $i$  and  $j$ .  $S$  and  $V$  are the salvation parameters and atom volume respectively and  $\sigma$  is set to 3.5 Å.  $r_{ij}$  is the interatomic distance between atoms  $i$  and  $j$  and  $s(r_{ij})$  is a smoothing function.  $E(t)$  is a function which provides directionality for the hydrogen bond term based on the angle  $t$ .  $\epsilon(r_{ij})$  is a dielectric function of  $r_{ij}$ .  $N_{rot}$  is the number of torsions in the receptor in its bound state.

The weights  $W_{vdW}$ ,  $W_{hbond}$ ,  $W_{elec}$ ,  $W_{desolv}$  and  $W_{rot}$  were empirically determined using linear regression on a set of ligand-receptor complexes with known binding constants.

The Lamarckian genetic algorithm mentioned above attempts to find the global minimum of equation 3.1 through conformational search. This release of binding free-energy was then returned as the ligand’s DS, which served as an estimate of its BA. This DS determined the reward to the agent.

A single docking calculation involves the evaluation of between  $10^6$  and  $10^8$  scoring function evaluations during the course of the genetic algorithm, which accounts for more than 90% of the algorithm’s runtime. Autodock-GPU [Santos-Martins et al., 2021] dramatically accelerates the runtime by exploiting the parallel nature of the docking algorithm, decomposing the population from a single generation of the algorithm into its individuals and assigning them to work-groups with each work group running on a parallel GPU compute unit.

### 3.3 Rewards and Shaping

Under the RL method, the state of the environment is evaluated by the reward function on every step to determine if the goal state has been reached and reward the agent accordingly. When treating drug design as a RL problem, the state of the environment is defined by the current molecular graph. However, the goal is that the agent discovers a novel molecular graph with high

DS for the given target **receptor**, therefore we cannot specify the goal molecular graph to the agent. Instead, a property that the goal state should possess is specified, and the reward function is defined accordingly. Here, the desired property is high DS for the specified **receptor** site of a given target molecule. Therefore, the reward simply returns the result of conformational search and the DS calculation performed by the docking package to the agent with its sign inverted, that is:

$$r(s_t) = -1 \times \Delta \text{ Binding Free Energy} \quad (3.3)$$

The reason for the inversion is that a greater reduction in binding free energy is of higher value. As a result, the q-value, or state-action value, of any molecular graph returned by the agent’s behavioural network, takes on the meaning of the expected cumulative DS expected to be obtained in the remaining steps by choosing that action.

Unlike in MolDQN [Zhou et al., 2019] which rewarded each state independently, which is possible since the calculation of properties such as QED and LogP are not computationally demanding, here it was not possible to calculate the DS of each transition along a rollout as this would have made the runtime of a complete training session infeasible. Therefore, it was only possible to calculate the DS of the final state along a rollout. Given that a rollout trajectory was composed of 40 transitions, only one of which receives an extrinsic reward, the problem was one of sparse rewards, and a shaping method was implemented to encourage learning.

### Transition Shaping

Given the reward for the terminal state of a rollout trajectory, rewards for the preceding transitions were calculated using a method similar to that of potential-based reward shaping [Ng et al., 1999]. Here the potential of the terminal state was taken to be the DS received from the environment, and the potentials of the preceding states were estimated by linearly interpolating from the full DS in the final state to zero in the initial state with the function

$$\phi(s_t) = r_f - \frac{r_f}{t_{max}} \times (t_i - 1) \quad (3.4)$$

where  $r_f$  is the reward of the terminal state,  $t_{max}$  is the number of steps in a trajectory and  $t_i$  is the steps remaining between state  $i$  and the terminal state.

## 3.4 Learning Algorithm

The RL algorithm (shown in Algorithm 1) used to train the agent was an implementation of double deep Q-networks (DDQN) [Mnih et al., 2013] [Mnih et al., 2015]. This section unpacks the algorithm in further detail. The line numbers that follow refer to Algorithm 1. First the agent was initialized with an empty molecular graph (line 4, Algorithm 1). The agent was then allowed a maximum of 40 steps in which to construct a **ligand**. The limit was chosen to be 40 as has been done in similar work [Zhou et al., 2019] [You et al., 2018] as it is necessary to define a limit to the length of a rollout episode, and since each step corresponds to the potential addition or removal of an atom or bond, this limits the maximum size of the **ligands** to 40 atoms, excluding hydrogen, with a mass in the range of 500 to 1000 Daltons, which is common for small-molecule drugs [Chhabra, 2021]. At each step of the modification phase, the environment generates all possible and valid molecular graphs that are accessible from the current state and returns these to the agent in a tensor. Note that in algorithm 1 the next states  $s_{t+1}$  are the actions  $a_t$ . Therefore, at each step of an episode the action space is defined by the accessible states. The agent then uses its behavioural network to select a state (action) to move into next (lines 5, 6, and 7, Algorithm 1). At each step, docking between the **ligand** and target **receptor** was simulated. The DS was calculated and returned to the agent as a reward ( $r_t$  in line 7, Algorithm 1), and the transition ( $s_t, s_{t+1}, r_t, term$ ) was stored in the agent’s memory (line 9, Algorithm 1), where *term* is a flag indicating if  $s_{t+1}$  was

**Algorithm 1:** Double Deep Q-Learning (DDQN) [Mnih et al., 2013] [Mnih et al., 2015]

```

1 Randomly initialize action-value function  $Q_\theta$  and target network  $Q_{\theta^-}$  with parameters  $\vec{\theta}$ 
   and  $\vec{\theta}^-$  respectively.
2 Initialise replay memory  $\mathcal{D}$  to capacity  $\mathcal{N}$ 
3 repeat
4   Reset environment
5   repeat
6     With probability  $\epsilon$  sample random action (next state)  $s_{t+1}$ 
7     Otherwise select  $s_{t+1} = \max_{a'} Q_\theta^*(s_t, s_{t+1})$ 
8     Move into next state  $s_{t+1}$  and observe reward  $r_t$ 
9     Store transition  $(s_t, s_{t+1}, r_t, term)$  in  $\mathcal{D}$ 
10  until  $t_{max}$ 
11  if  $e \bmod T_{bp} == 0$  then optimize behavioural and target networks
12    Sample random minibatch of transitions  $(s_j, s_{j+1}, r_j, term)$  from  $\mathcal{D}$ 
13    Set  $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{t+1}, a') & \text{for non-terminal } s_{j+1} \end{cases}$ 
14    Update  $Q_\theta$  via one step of gradient descent by backpropagation
15     $\Delta \vec{\theta} = \nabla_\theta J(\theta) = \mathbb{E} \left[ \left( r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a) \right) \nabla_\theta Q_\theta(s, a) \right]$ 
16     $\theta^- \leftarrow \tau \theta$  // soft update target network
17  end
18 until  $e_{max}$ 

```

a terminal state. A trajectory of at most 40 such transitions ( $t_{max}$  in algorithm 1) constituted a single rollout or exploration episode.

After  $T_{bp}$  rollout episodes (the backpropagation period) where data was sampled from the policy  $\pi_\theta$  the agent randomly sampled a minibatch of transitions from the replay memory (line 12, Algorithm 1) and optimized its ANN approximation of the optimal action-value function  $Q_\theta$  through stochastic gradient descent using the backpropagation algorithm (lines 13, 14 and 15, Algorithm 1). The parameters of the target network  $\theta^-$  were then updated with a fraction  $\tau$  of the updates made to their counterparts in the behavioural network (line 16, Algorithm 1).

Thus training comprised an alternation between  $T_{bp}$  sampling episodes  $e$  and minibatch gradient descent through backpropagation. This loop continued for  $e_{max}$  episodes or until convergence was observed in the policy network's loss function.

## Double Deep Q-Networks

The objective for deep Q-learning is to train an ANN to approximate the optimal action-value function

$$Q^*(s, a) = \max_{\pi} \mathbb{E} \left[ R_t = \sum_{i=t}^T \gamma^{i-t} r_i \mid s_t = s, a_t = a \right] \quad (3.5)$$

The optimal action-value function  $Q^*(s, a)$  approximates the maximum expected return  $R_t$  obtainable from state  $s$  after taking action  $a$  and following policy  $\pi$  thereafter.  $R_t$  corresponds to the maximum sum of rewards  $r_t$  obtainable by a policy  $\pi_\theta$  from the start state discounted by  $\gamma$  at each time step.

Q-learning [Mnih et al., 2013] introduced the biologically inspired process of experience replay.



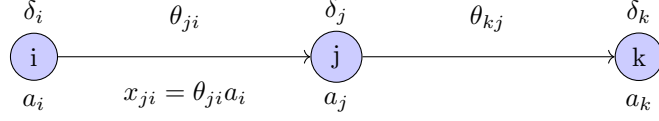


Figure 3.3: Schematic explaining the notation used in the backpropagation algorithm

This improved sample efficiency, since each experience is potentially used in many weight updates. It also removed correlations between samples in the minibatches used for backpropagation since each minibatch contains a random sampling of transitions from multiple trajectories. Removing such correlations reduced the variance caused by learning directly from consecutive samples, which could lead to premature suboptimal convergence. The DDQN variant [Mnih et al., 2015] introduced a second network, which decoupled the behavioural policy used to sample trajectories from the network which approximated target values  $r + \gamma Q(s', a')$  during training. This decoupling reduced correlations between the rollout and target networks which could also lead to divergence during training.

The optimal action-value function given in equation 3.5 obeys the Bellman equation, which is an identity between the optimal action-value at state  $s$  and the optimal action-value of the subsequent state  $s'$ . The Bellman equation states that

$$Q_{i+1}^*(s, a) = \mathbb{E}_{s' \sim \mathcal{E}} [r + \gamma \max_{a'} Q_i^*(s', a') \mid s, a]. \quad (3.6)$$

That is, the optimal value of taking action  $a$  in state  $s$  is equivalent to the reward for moving into state  $s'$  plus the maximum action-value obtainable in state  $s'$ . The method uses a deep ANN to approximate the optimal action-value function  $Q_\theta(s, a)$  and a memory buffer  $D_t = [e_1, \dots, e_t]$  to store the transitions  $e_t = (s_t, a_t, r_t, s_{t+1})$  obtained from rollout trajectories. The objective of learning is approximate the optimal action-value function by using equation 3.6 as an iterative update. This is a form of value iteration which converges to the optimal value function  $Q_i \rightarrow Q^*$  as  $i \rightarrow \infty$ .

Specifically, the ANN approximating  $Q^*$  is trained by minimizing the loss function

$$J_i(\theta) = \mathbb{E}_{(s, a, r, s') \sim U(D)} \left[ \left( r + \gamma \max_{a'} Q_{\theta_i^-}(s', a') - Q_{\theta_i}(s, a) \right)^2 \right] \quad (3.7)$$

through sampling minibatches of transitions randomly from the replay memory  $(s, a, r, s') \sim U(D)$ .  $Q_{\theta_i}$  is the approximation of the action-value function with the parameters as they were at iteration  $i$  and  $Q_{\theta_i^-}$  is the target network at iteration  $i$ , which is the approximation of the action-value function with parameters  $\theta_i^-$  where  $\theta_i^- \leftarrow \theta_i$  every  $C$  steps.

Gradient descent is then effected through backpropagation, which locates the parameters  $\vec{\theta}$  which minimize the loss function  $J(\theta)$  by calculating an update  $\Delta\theta$  for every parameter in the network

$$\vec{\theta} \leftarrow \vec{\theta} + \eta \Delta\vec{\theta}. \quad (3.8)$$

The update to apply to each parameter in the ANN at every iteration of backpropagation is the gradient of the loss function

$$\Delta\vec{\theta} = \nabla_{\theta} J(\vec{\theta}) \quad (3.9)$$

where  $\eta$  is the learning rate. The gradient of the loss function with respect to the network parameters  $\nabla_{\theta} J(\vec{\theta})$  is

$$\nabla_{\theta} J(\vec{\theta}) = \mathbb{E} \left[ \left( r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_{\theta}(s, a) \right) \nabla_{\theta} Q_{\theta}(s, a) \right], \quad (3.10)$$

and the explicit update for parameter  $\theta_i$  is

$$\theta_{ji} \leftarrow \theta_{ji} + \eta \frac{\partial J(\theta)}{\partial \theta_{ji}}. \quad (3.11)$$

This update for each parameter  $\theta_i$  is obtained with the backpropagation algorithm [Rumelhart et al., 1986]. For some hidden parameter  $\theta_{ji}$  with input neuron  $i$  and output neuron  $j$ , the gradient of the loss function with respect to  $\theta_{ji}$ ,  $\frac{\partial J(\theta)}{\partial \theta_{ji}}$ , is given by the identity

$$\Delta \theta_{ji} = \frac{\partial J(\theta)}{\partial \theta_{ji}} = \delta_j a_i. \quad (3.12)$$

Where  $\delta_j$  is the gradient of the loss function at neuron  $j$  and  $a_i$  is the neuron activation at neuron  $i$ . For an output neuron  $k$ , it's gradient  $\delta_{k,o}$  is given by

$$\delta_k = \frac{\partial J}{\partial a_k} \frac{\partial a_k}{\partial net_k} \quad (3.13)$$

where  $a_k$  is the activation of neuron  $k$  and  $net_k = \sum_h x_{kh} = \theta_{kh} a_h$  is the net input into neuron  $k$ . For a hidden neuron  $j$ , it's gradient  $\delta_j$  is given by

$$\delta_j = \frac{\partial a_j}{\partial net_j} \sum_k \delta_k \theta_{kj}. \quad (3.14)$$

Where  $\sum_k \delta_k \theta_{kj}$  is the sum over the products of the gradients of  $j$ 's output neuron activations and their respective parameters  $\theta_{kj}$ . The relationship between neurons, their activations  $a$ , synapses  $\theta$  and gradients  $\delta$  is shown schematically in figure 3.3.

### 3.5 Summary

This chapter presented the methods implemented in this research pursuant to the goal of automating the design of **antagonistic ligands** for a **receptor** with a known crystal structure. The chapter discussed the three facets of the generative framework design, namely the agent, the environment and the learning algorithm. In summary, the agent was implemented as a pair of coupled ANNs and memory buffer, which perceived the molecular graphs that defined the environment states as extended-connectivity fingerprints. The environment was composed of the **ligand** graph, the **receptor** model, docking algorithm, reward function and reward shaping method. The learning algorithm used was a neural implementation of Q-learning, specifically the DDQN variant.

The learning process is analogous to drawing graphs by hand, at each step adding or removing bonds or atoms, keeping in mind the goal that the resulting graph should have a high **DS** for the target **receptor**, and learning from the feedback received about each resulting graph. The DDQN algorithm makes it possible for the agent to quickly find a path through chemical space to graphs which receive high rewards, and as a result, the output is a number of graphs with high expected **BA** for the target **receptor**.

In order to evaluate the generative algorithm, it was applied to **ATP** binding site of the **PfPI4K enzyme** homology model, shown in figure 3.1, under a number of case-studies that are explored in the following chapter.

## Chapter 4

# Experiments and Results

In this chapter the results of training the agent’s optimization policy against the chosen target [receptor](#) are presented. The target [receptor](#) of interest was the [ATP](#) binding site of the human [PfPI4K enzyme](#), shown in [figure 3.1](#). The agent was trained for a total of 5,000 episodes to limit the total run-time. However, training for longer may have yielded further improvement than what is reported here. Training was tracked through the use of plots of batch loss and reward per episode, where the reward per transition depended on the experiment being run. The agent was trained with the [DDQN](#) algorithm presented in [Algorithm 1](#) using the environment parameters, network hyperparameters and the dimensions and location of the docking box given in [table A.1](#).

The experiments were chosen in order to answer the research questions posed in [section 1.2](#). In summary, the experiments conducted in this chapter include:

- The generation of molecules with maximal [QED](#) starting from a single carbon atom ([section 4.1](#)).
- The generation of [ligands](#) with maximal [DS](#) for a target [receptor](#) when rewarding only the terminal state in a rollout episode ([section 4.2.1](#)).
- The generation of [ligands](#) with maximal [DS](#) for a target [receptor](#) with the use of reward shaping to distribute the terminal state reward across the preceding transitions in the rollout ([section 4.2.2](#)).
- The algorithm composed of shaped sparse [DS](#) rewards was run starting from three different [chemotypes](#): naphthyridine, imidazopyridazine, and aminopyradine to assess the impact of starting from a region of chemical space known to contain structures with high [DS](#) for a target ([section 4.2.3](#)).
- Finally, a set of 1,011 structural analogues of naphthyridine, imidazopyridazine, and aminopyradine were docked with the target [receptor](#) and their [DSs](#) were used as a reference for comparing the scores of the [ligands](#) generated by the agent. Their [DSs](#) were also plotted as a function of their empirical [half maximal inhibitory concentration \(IC50\)](#) measurements to assess the correlation between [DS](#) and real-world [ligand](#) potency ([section 4.2.4](#)).

The [QED](#) and [DS](#) maximization experiments address research question 1: *what are the methodological requirements for applying RL to the generative design of ligands for drug targets?* The compilation of the [DSs](#) and [IC50](#) values of the reference [ligands](#) and their comparison with those produced by the agent address research question 2: *how do the generated ligands compare those currently available?* By considering the results of these experiments together we can address research question 3: *is this method viable?*

The results of these experiments in relation to the research questions are discussed in the following chapter.

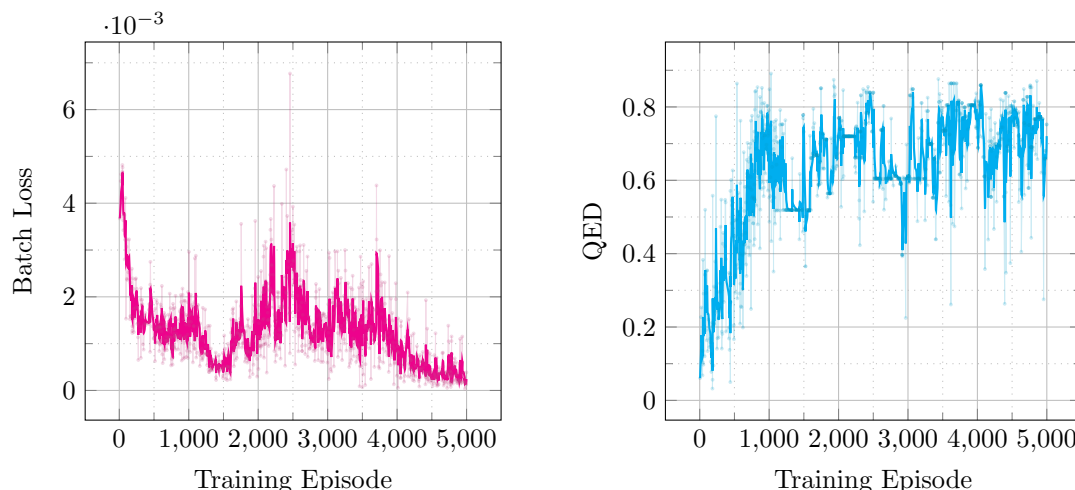


Figure 4.1: Training curves from the dense QED rewards experiment showing mean batch error (left) and QED of the terminal state (right) per training episode. The data are the original values passed through a first-order IIR low-pass filter  $f(x_t) = \alpha x_{t-1} + (1 - \alpha)x_t$  with  $\alpha = 0.6$  to attenuate the high-frequency components and emphasize the trend.

## 4.1 QED Optimization

As a benchmark of the DDQN implementation and molecule environment the framework was first applied to the maximization of QED, that is a replication of one of the results from the original MolDQN paper [Zhou et al., 2019].

QED corresponds to the weighted geometric mean of eight desirability functions (equation 4.1) [Bickerton et al., 2012]

$$QED = \exp\left(\frac{\sum_{i=1}^n w_i \ln d_i}{\sum_{i=1}^n w_i}\right) \quad (4.1)$$

where  $d_i$  is the  $i$ th desirability function and  $w_i$  is the weight applied to the corresponding desirability function. The desirability functions are molecular weight, octanol-water partition coefficient, number of hydrogen bond donors, number of hydrogen bond acceptors, molecular polar surface area, number of rotatable bonds, number of aromatic rings, and the number of structural alerts.

The goal here was for the agent to learn a policy for choosing a sequence of actions which leads to molecules with maximal QED. This was intended to verify the implementation before extending the framework to the more challenging task of targeted ligand generation.

Given that the learning algorithm is stochastic, every training session will yield a different set of molecules, therefore a given training session was evaluated by the QED value on which the policy converged and the three molecules with the highest QED generated over the course of training.

In this experiment the learning task was the simplest, and the agent received a reward on every step of a rollout episode where the reward was the QED of the chosen state. The reward was further scaled by the steps remaining from the terminal state (equation 4.2)

$$s(r_t) = r_t \times \alpha^{t_{max} - t} \quad (4.2)$$

such that states earlier in a rollout episode have their reward reduced, so that the agent has to obtain high-value states near the end of the episode in order to maximize its reward. The training curves from this experiment are given in figure 4.1.

The plot in figure 4.1 shows the scaled reward for the terminal state in each 40-step rollout episode. These rewards were the QED score scaled with equation 4.2 and consequently the actual QED scores of the generated molecules were slightly higher than the rewards plotted in figure 4.1.

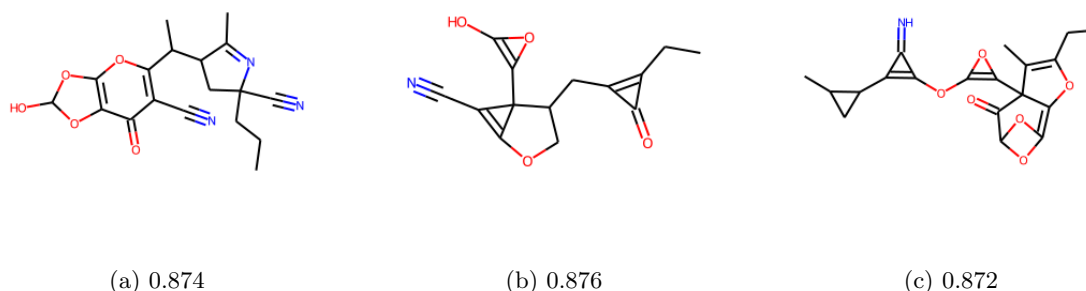


Figure 4.2: Three molecules generated by the agent during the dense QED rewards experiment. The goal here was only to maximize QED. The terminal state QED reward is shown below each graph, a: 0.874, b: 0.876 and c: 0.872. The QED score here was the score returned from the RDKit QED package scaled with equation 4.2 so the actual QED value was slightly higher.

## 4.2 Ligand Generation

Here the framework was extended to the more challenging task of generating ligands with maximal BA for the receptor site of a macromolecule, as approximated by their Autodock DS, defined in equation 3.1, and measured in kcal/mol.

As in the QED experiment, the agent’s goal was to construct ligands through the addition or removal of atoms or bonds, such that the terminal state after 40 transitions received a high DS, subject to no other constraints. When the environment was reset at the beginning of each new episode the state of the ligand was returned to its initial state.

Two of the experiments explored the impact of the initial state on the final DS on which the agent converged. In section 4.2.2 the agent begins from a single carbon atom, whereas in section 4.2.3 the agent begins from one of three reference scaffolds.

And two of the experiments contrast the impact of sparse rewards on the agent’s performance. In section 4.2.1 only the terminal state receives a DS reward whereas in section 4.2.2 reward shaping is used to estimate a reward for all transitions in the rollout episode.

The target receptor for these experiments was the ATP binding site of the PfPI4K homology model [Fienberg et al., 2020] shown in figure 3.1 and all docking simulations used the docking box as defined in table A.1.

The performance of the algorithm was evaluated by the DS on which the agent converged after 5000 training episodes, and the highest DSs achieved during the training session.

### 4.2.1 Sparse Rewards from Docking Simulations

When rewarding the agent with a ligand’s DS, it was not possible to return a reward at every step of a rollout episode as the time taken to calculate a single ligand’s DS would make the runtime for a full training session of 5000 episodes untenable. Therefore, a DS was returned only for the terminal state of the rollout episode. This meant that the preceding transitions were added to the replay memory with no associated reward. The purpose of this first experiment was to evaluate whether it was possible for the agent to achieve some degree of success with such sparse feedback from the environment. The training curve showing terminal state DS for this experiment is given in figure 4.3.

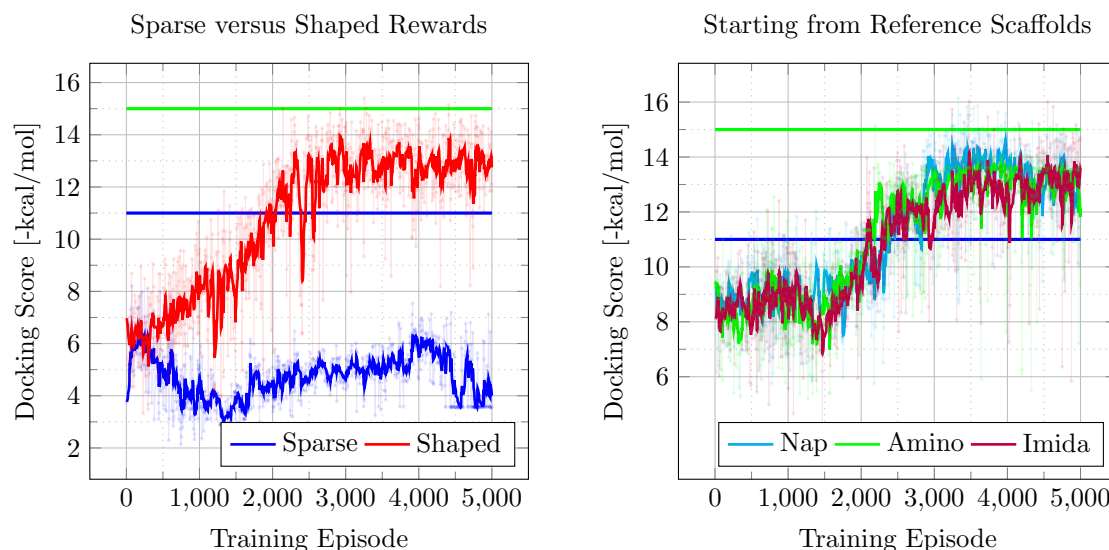


Figure 4.3: Training curves from the various DS rewards experiments showing the Autodock DS of the terminal state per training episode. The plots show the raw data as well as a that data smoothed with the following first-order IIR low-pass filter  $f(x_t) = x_{t-1} * \alpha + (1 - \alpha)x_t$  with  $\alpha = 0.8$ . The data show the agent’s performance when learning from sparse rewards (left), using reward shaping (left), and starting from a number of known reference molecules (right). The abbreviations in the legend in the figure on the right correspond to the reference scaffolds naphthyridine, aminopyridine and imidazopyridazine. The experiments starting from the reference molecules also incorporated reward shaping. The two horizontal lines show the mean and maximum DS of the ligands in the reference set shown in figure 4.5.

#### 4.2.2 Docking Simulations with Reward Shaping

The next experiment attempted to create dense reward signals from the sparse extrinsic reward returned by the environment at the end of the agent’s rollout trajectory.

This was done with equation 3.4. Here, instead of pushing the states received from the environment along with a reward of zero to the agent’s replay memory as they were received, they were instead buffered until the end of the episode whereupon equation 3.4 was used to calculate a non-zero reward for every preceding transition using the reward received for the terminal state.

The agent was trained for 5,000 episodes with the parameters given in table A.1. The training results are shown in figure 4.3 alongside the results from the previous sparse-rewards experiment.

#### 4.2.3 Generation from Reference Series

As a final experiment, the effect of focusing the search on regions of chemical space near to the structural features of chemical series known to have an affinity for the receptor was explored. Here the variant of the algorithm incorporating reward shaping was applied to three runs, each composed of 5000 episodes starting from one of the three scaffolds shown in figure 4.6. These scaffolds are the aminopyridine, naphthyridine and imidazopyridazine molecular backbones. These chemical series are being investigated as derivatives of these three molecular backbones with varying substituents have demonstrated high inhibition potency against the target enzyme in phenotypic whole-cell screenings. The plots of terminal state DS over the course of training for these three runs are shown in the right sub-figure of figure 4.3.

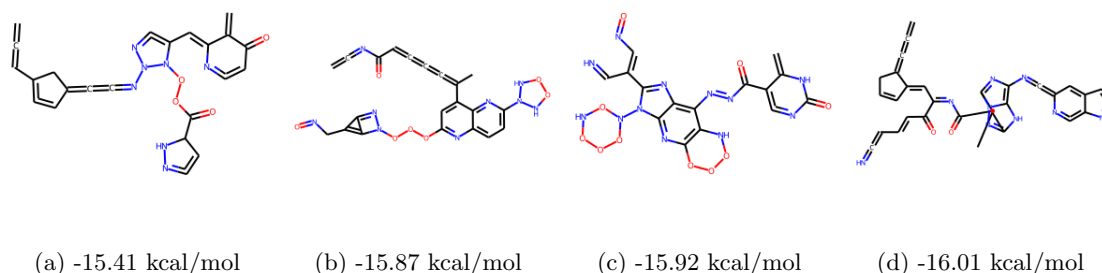


Figure 4.4: ligands and their DSs from the following experiments: (a) reward shaping starting from a single carbon with a DS of -15.41 kcal/mol, (b) starting from naphthyridine with a DS of -15.87 kcal/mol, (c) starting from aminopyridine with a DS of -15.92 kcal/mol, and (d) starting from imidazopyridazine with a DS of -16.01 kcal/mol. Note that the accuracy of the DSs is limited by the number of genetic algorithm runs.

#### 4.2.4 Comparison with Existing Ligands

In order to evaluate the algorithm’s performance when generating ligands for the chosen receptor, a reference point was needed. Therefore, a set of ligands with experimentally measured IC<sub>50</sub> values were docked against the target receptor for comparison. An IC<sub>50</sub> measurement is the concentration of a chemical species necessary to induce a 50% reduction in the biological activity of a target enzyme or protein, where the measurements are obtained through phenotypic whole-cell screenings. These are *in vitro* bioassays where a solution of the ligand is added to a sample of human cells infected with the target enzyme and the reduction in enzymatic activity is typically measured through some form of spectroscopy.

The ligands and their IC<sub>50</sub> data were provided by the UCT H3D drug design lab <sup>1</sup>, and they contained structures from the naphthyridine, imidazopyridazine (SFK52), and aminopyridine (SFK40) chemical series. These series contain ligands derived from the scaffolds shown in figure 4.6. Each ligand was docked using the parameters given in table A.1. The DSs of the reference ligands are plotted as a histogram in figure 4.5 (left), and they are plotted as a function of their empirical IC<sub>50</sub> data in figure 4.5 (right).

The DSs in figure 4.5 (right) are plotted as a function of pIC<sub>50</sub> (equation 4.3)

$$\text{pIC}_{50} = -\log(\text{IC}_{50}[\text{nM}] \times 10^{-9}) \quad (4.3)$$

This is done due to the significant range spanned by IC<sub>50</sub> measurements which makes it necessary to constrain them to a logarithmic scale in order to facilitate a correlation plot.

### 4.3 Summary

The experiments presented in this chapter were chosen to answer the three research questions posed in section 1.2. In summary, the goal was for the experiments to address whether, given the crystal structure of a receptor, (1) RL is a viable method of generating novel ligands with high DS, (2) how the DSs and structures of those ligands compare to known ligands, and (3) what were the methodological requirements of applying the method in this way.

As a baseline, section 4.1 evaluated the DDQN agent’s ability to return molecular graphs with high QED. Section 4.2 then extended the DDQN agent to the harder problem of returning

<sup>1</sup><http://www.h3d.uct.ac.za/>

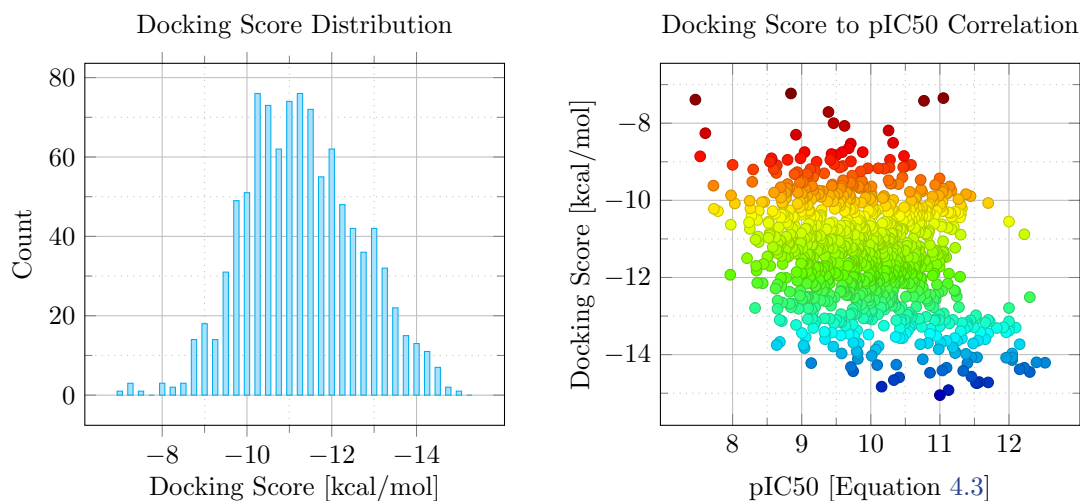


Figure 4.5: Results of docking a set of 1,011 structural analogues derived from naphthyridine, aminopyridine and imidazopyridazine scaffolds (figure 4.6) against the PfPI4K receptor. **Left:** the  $x$ -axis corresponds to DS intervals of 0.25 kcal/mol and the  $y$ -axis corresponds to the number of ligands in each interval. This shows the range and distribution of Autodock DSs for known ligands, which provides a benchmark for the agent’s performance. **Right:** the  $y$ -axis corresponds to DS in kcal/mol and the  $x$ -axis corresponds to pIC50 (equation 4.3). This evaluates the correlation between Autodock DSs and empirical IC50 measurements and gives an estimate of the confidence with which we can expect the DSs to correlate with activity *in vitro*.

molecular graphs with high expected BA, estimated through their Autodock DS, for the specified receptor region of a given crystal structure.

Within section 4.2, the agent first received a reward at only the terminal state of a design episode (section 4.2.1). Reward shaping was then used to create dense rewards from these sparse extrinsic rewards (section 4.2.2). This variant with sparse extrinsic rewards and reward shaping was then applied to the modification of three chemotype scaffolds (section 4.2.3).

Finally, as a benchmark for the algorithm, the DSs of a set of 1,011 known ligands from the same chemical series as the scaffolds in section 4.2.3 were calculated and plotted, as a histogram and correlation plot with their pIC50 measurements (section 4.2.4).

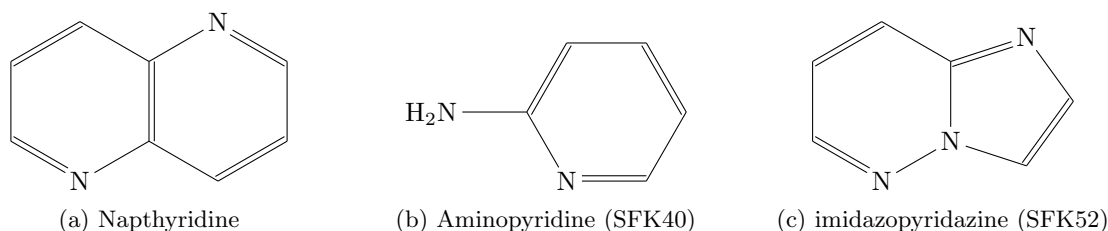


Figure 4.6: The three scaffolds from which the 1,011 structural analogues evaluated in figure 4.5 were derived. Each ligand contained either the naphthyridine (a), aminopyridine (b) or imidazopyridazine (c) backbone with varying substituents. All ligands were docked with the PfPI4K homology model using the same docking parameters as were used in the preceding experiments (table A.1).



# Chapter 5

## Discussion

This chapter elaborates on the results presented in chapter 4 by summarizing them, providing an analysis of *why* the experiments elicited the specific results that they did, and stating any implications which the results could have for the current state-of-the-art in the field of automatic drug design through ML. These supporting explanations make use of references to previous related work where possible.

The topics discussed include the framework’s baseline performance on the QED optimization problem, the impact of sparse versus shaped rewards, the impact of beginning from a single atom versus a reference scaffold, and a comparison between the DSs of the generated ligands versus those of the reference set. The chapter ends by tying the results back to the research questions outlined in section 1.2.

### QED Optimization

The results of the QED optimization experiments, shown in figure 4.1, demonstrated that the agent is able to rapidly learn a policy for selecting molecular fingerprints in order to maximize a molecular property such as QED. This is evidenced by the fact that the QED score of the terminal state began at less than 0.1 at the beginning of training, when the agent was choosing actions almost entirely at random, and quickly converged within 1000 attempts to a value greater than 0.8. It is important to note that the values plotted in figure 4.1 correspond to the terminal state rewards, which were the QED scores scaled by equation 4.2, and consequently the actual QED scores were actually slightly higher than those shown in figure 4.1. Three of the molecules generated from this experiment, together with their terminal state QED scores, are shown in figure 4.2.

This result was expected as it has already been shown [Zhou et al., 2019] that the DDQN algorithm is capable of producing good results on the QED optimization problem. Furthermore, the problem is significantly simpler than that of targeted ligand generation as the subset of chemical space containing molecules with high QED scores is orders of magnitude larger than that containing molecules with high DSs for the receptor site of a particular crystal structure. In addition to this, the agent receives, on every step of a rollout episode, a QED score for the state that was chosen at that step. This provides the agent with ample samples from which to correlate the features of an extended connectivity fingerprint with high QED.

Thus, the primary utility of this result is to confirm that the framework was working as expected before extending it to the harder problem. However, this result also confirms that of [Zhou et al., 2019] by showing again that DDQN can easily be applied to maximize simple physico-chemical properties like QED which in itself is potentially very useful for drug design.

## Sparse Rewards and Shaped Rewards

The initial generation experiment where the agent received a reward only for the terminal state demonstrated that this is insufficient information from which to extract a useful policy as the DS of the generated ligands failed to increase within 5,000 episodes. The terminal state DS after 4,500 training episodes was actually lower than the DSs of terminal states obtained from almost entirely random exploration within the first 500 episodes.

However, a simple reward shaping method of linearly extrapolating backwards from the final state, to calculate a reward for the preceding states using equation 3.4, immediately enabled the agent to learn an effective design policy. The “shaped” plot in the left axis of figure 4.3 shows that after starting at an initial DS of greater than -6 kcal/mol, obtained from random exploration, the agent converged on a minimum of approximately -13 kcal/mol with the best candidates exceeding -15 kcal/mol.

The effectiveness of this shaping methods is likely due to the fact that the value of a given state is not determined by the state in isolation, but the terminal state of the path in which the state occurred, as it is the terminal state that is returned by the agent. That is to say, states are valuable if they are close to terminal states with high DSs. For example, if a given rollout episode resulted in a terminal state which received a high DS, then the penultimate state along that path would also be valuable as it permits access to the high-scoring terminal state. The further back along the path you move away from the terminal state, the less valuable the states become. The shaping function (equation 3.4) was designed to implement this logic, thus conditioning the agent to select features in the extended connectivity fingerprints which will lead to high-scoring terminal states within the 40-step limit.

Given that the use of reward shaping enabled the agent to learn where it had previously failed in the sparse reward environment, we can assume that a more sophisticated sparse-reward amelioration strategy such as hindsight experience replay, which has been shown experimentally to outperform reward-shaping [Andrychowicz et al., 2017], should improve the algorithm’s performance.

## Generation from Reference Ligands

The left-hand side plot of figure 4.3 shows the results when the agent begins building from a single carbon atom. These experiments evaluated the agent’s ability to find a path from an arbitrary point in chemical space to a region of high DS for the receptor.

The plot on the right-hand side of figure 4.3 shows the results when the agent begins its rollout episodes from a scaffold known to be a structural analogue of high-DS ligands. Here the agent began from naphthyridine, aminopyradine, and imidazopyridazine, shown in figure 4.6.

The corresponding plots in figure 4.3 show that when starting a 40-step rollout from each of these scaffolds the initial ligands generated by the agent from random exploration obtained a better terminal state DS than those obtained when beginning from only a single carbon atom as the terminal state DSs from random rollouts starting from naphthyridine, aminopyradine and imidazopyridazine were approximately -9 kcal/mol.

After approximately 2,500 training episodes the agent converged on candidate ligands with an average terminal state DS of approximately -14 kcal/mol, with numerous candidates exceeding -15 kcal/mol and the best exceeding -16 kcal/mol. This is comparable to the performance of MolDQN [Zhou et al., 2019] which converged on a score of 0.8 after approximately 3,000 episodes when training to optimize QED.

In summary, when starting from scaffolds known to be structural analogues of ligands with high DS for the target, the agent begun training by finding, through initial random exploration, states with a better DSs than those discovered when beginning from only a single carbon atom, and

converged on states with better *DSs* than those obtained when starting from only a single carbon atom.

The reason that the agent converges on better molecules when building from known *scaffolds* is likely that the search is now being constrained to the region of chemical space surrounding the given *scaffold*. And since the aminopyradine, naphthyridine and imidazopyridazine *scaffolds* are the backbones of three chemical series known to contain high-*BA ligands*, by constraining the search to this region, the problem of finding high-*BA* candidates is simplified since the agent is more likely to discover rewarding states through exploration.

This is expected since structural analogues, molecules which share similar *scaffolds* but with different substituents and a sufficiently similar molecular backbone, are candidates for *functional analogues*, which are two molecules with similar pharmacological properties. That is, they exhibit similar biochemical or physiological effects on the human body, but with potential variations in efficacy and side effects [Bruice, 2011].

Furthermore, this is supported by how many *EAs* use specially pre-initialised populations to boost the task performance of evolved solutions by evolving novel *functional analogues* of the initial population [Rupakheti et al., 2015, Brown et al., 2004a, Lameijer et al., 2005]. In other RL approaches the network has also been initially pre-trained on a set of structural analogues [Sumita et al., 2018].

The notion of structural analogues has also been incorporated in RL algorithms through the use of Tanimoto similarity [Zhou et al., 2019], where the agent attempts to maximize similarity to a given *scaffold* when generating novel structure.

## Comparison with Known Ligands

Figure 4.5 shows the *DSs* and experimentally measured *IC50* data of 1,011 structural analogues of the naphthyridine, aminopyradine and imidazopyridazine *scaffolds*. These structures were docked with the same parameters used to reward the agent during the generation experiments. From the histogram we see that the mean of the set is approximately -11 kcal/mol with only a very small number of structures receiving scores less than -14 kcal/mol with none being below -16 kcal/mol.

Furthermore, the scatter plot of *DS* versus *pIC50* measurements in figure 4.5 demonstrates a weak correlation between *DS* and *IC50*. The weakness of the correlation could be attributed to inaccuracies in the Autodock scoring function given that rigid bodies are used for both the *receptor* and *ligand* models and no solvent interactions are considered. The docking simulation also does not consider off-target interactions with *endogenous* biomolecules, which could contribute to the *IC50* readings as they are obtained from whole-cell phenotypic screenings. It is also important to note that, given the large number of *ligands* being docked, in order to reduce the total time taken to calculate the *DSs* for all 1,011 *ligands*, docking was only run for 10 epochs which could yield suboptimal accuracy. This is because after each epoch the docking algorithm returns the *DS* of the lowest energy conformer. However, due to the stochastic nature of the *EA* which searches conformer space, the algorithm is not guaranteed to find the lowest energy conformer in a single epoch. And so it is recommended that multiple epochs are run, with the *DS* taken to be the lowest of the epochs. Limiting the number of epochs to 10 could potentially prevent the search algorithm from finding the lowest-energy conformer of a given *ligand*.

Despite this, the scatter plot does show that those *ligands* with the best *DSs* of less than -14 kcal/mol also have low *pIC50* values. Furthermore, the *ligands* with the worst *DSs* of greater than -9 kcal/mol, do in general have *pIC50* values greater than those of the best *ligands*. This indicates that the Autodock scoring function is able to identify features which correlate with very low or very high *pIC50*, but for other features it struggles to accurately correlate them with *pIC50*.

So in summary, the results in figure 4.3 show that the agent was able to design a set of *ligand* candidates which contains a substantial number of *ligands* with *DSs* better than the best *DSs* in a set of reference *ligands*. The comparison with the *IC50* data shows that within the accuracy of the

Autodock scoring function, these **ligands** with high **DSs** can be expected to also potentially yield low **pIC50** values.

When inspecting the graphs generated by the agent, four of which are shown in figure 4.4, it is apparent that although the agent is able to improve **DS**, it does so without any concern for **QED** or **SA**, since these properties were not included in the reward function, and as a result the proposed molecules contain a significant amount of unsynthesizable features. This issue is discussed more in the following chapter.

The implication of these results is that the **DDQN** algorithm is effective at locating regions of chemical space which maximize the Autodock scoring function, at least for the **PfPI4K** structure. Therefore, if we are able to increase the correlation between **DSs** and **pIC50**, as well as improve the **QED** and **SA** of the generated graphs, we can expect a **DDQN** agent to reliably generate potent inhibitors.

## Answering the Research Questions

Regarding the methodological requirements for applying **RL** to the problem of **ligand** generation, as long as the reward received from the environment is dense, either because you are able to dock every transition or due to the use of some method of reward shaping, the **RL** method is more than capable of learning a policy for constructing a **ligand** through a sequence of atom and bond additions or removals such that the result receives a high **DS**.

Rewarding only the terminal state during training significantly reduces training time, and using a reward shaping method facilitates learning when doing so. However, other sparse rewards strategies such as hindsight experience replay might work better [Andrychowicz et al., 2017].

Furthermore, it is necessary to incorporate **QED** and potentially **SA**, as penalties into the reward function in order to filter out the unrealistic features which occur when **DS** is rewarded in isolation. This can also be accomplished by hard-coding filters for structural flags into the training environment.

In addition to this, starting generation from **scaffolds** known to be analogues of **ligands** with high **DSs** also appears to simplify the search problem for the agent thus leading to the policy converging on **ligands** with a higher **DSs**.

When comparing the artificially designed **ligands** with those currently available, we see that the agent is able to generate a substantial number of **ligands** with better **DSs** than the best **ligands** in the reference set. However, whether these artificially designed **ligands** are physically realistic is not guaranteed because even though they may be 100% valid according to the valence rules, they may still contain unrealizable structural features. Hence, the need for incorporating additional structural penalties into the reward function, and potentially hard-coding flags for impossible structures as well.

In addition to this, whether these **ligands** with high **DSs** can be expected to also yield high **IC50** measurements, which is how we would experimentally verify their potency, depends on the accuracy of the docking simulation. If we are able to improve the correlation between **DS** and real **ligand** potency, potentially through the use of full molecular dynamics simulations, the method has the potential to yield valuable results.

So in summary, the method of applying **RL** to the problem of automatic drug design is indeed viable, however this is subject to the constraints of removing unstable and unsynthesizable features from the generated graphs without sacrificing **DS**, and improving the correlation between **DS** and *in vitro* whole-cell **ligand** potency as measured by **IC50**.

## Chapter 6

# Conclusions

This research sought to investigate the potential for applying RL to the development of generative algorithms for automated drug design. The methods comprised the development of a RL agent and learning environment where the agent could construct molecular graphs bond-by-bond and dock the resulting ligands with the crystal structure of a target receptor. The experiments evaluated the agent and environment at the tasks of maximizing QED and DS for a given receptor. They explored the impact of sparse versus shaped rewards, focusing the search to a particular region of chemical space, and compared the ligands generated by the agent with those currently being evaluated in the laboratory.

The first experiment, the maximization of QED starting from a single carbon atom, served as a benchmark to confirm that the algorithm was successful at the simpler task. The second experiment was the generation of ligands with maximal DS for a target receptor when rewarding only the terminal state in a rollout episode. This demonstrated that while rewarding only the terminal state reduced the run-time compared to docking all states in a rollout, it prevented the agent from learning a successful policy due to insufficient feedback from the environment. The third experiment repeated the previous test with the use of reward shaping to distribute the terminal state reward across the preceding transitions in the rollout. This immediately facilitated learning. The fourth experiment applied the algorithm with reward shaping to three reference series: naphthyridine, imidazopyridazine, and aminopyridine. This demonstrated that starting from a point in chemical space known to contain ligands with high BA for the target, as opposed to starting from a single carbon atom, simplifies the problem for the agent thus yielding more ligands with better DSs. Finally, a set of structural analogues derived from the three reference scaffolds was docked with the target receptor to benchmark the expected DSs and evaluate the correlation between DS and IC50 measurements.

The QED problem was simple for the agent to solve as it was receiving dense extrinsic rewards, and the subset of chemical space containing molecules with high QED scores is orders of magnitude larger than that containing molecules with high DSs for the receptor site of a particular crystal structure. The reward shaping method was a success due to how the shaping function assumed the value of a state to be proportional to both the DS received by the terminal state of that episode and its distance from the terminal state. When beginning the training episodes from the reference scaffolds, the agent was able to discover more molecules with higher DSs due to the search space being focussed to the region of chemical space where the naphthyridine, aminopyridine and imidazopyridazine series are located. Finally, the comparison with the set of known ligands showed that agent was able to discover numerous molecules with DSs better than the best scores in the reference set. However, the graphs generated by the agent contained a substantial number of unsynthesizable features due to the fact that the reward function considered DS in isolation, with no consideration being made for QED or SA. This could likely be improved by building penalties

for these properties into the reward function. In addition to this, the weak correlation between *DS* and *pIC50* was likely due to the fact that rigid bodies were used for both the *receptor* and *ligand* models and no solvent interactions are considered. Furthermore, the docking simulation does not consider off-target interactions with *endogenous* biomolecules. This could likely be improved with the use of full molecular dynamics simulations.

So in conclusion, there is significant potential for applying *RL* to obtaining novel drug designs, however, for the method to be useful it is necessary to prevent the agent from building chemically unstable and unsynthesizable features into the generated graphs without sacrificing *DS*.

## 6.1 Future Work

The preceding discussions mentioned a few possible extensions to the framework developed in this thesis which could resolve various shortcomings identified in the current implementation. This section unpacks these in more detail.

The most apparent shortcoming of the current implementation is the number of impossible atomic arrangements which appear in the graphs generated by the agent, rendering the proposals unsynthesizable even though they possess high *DSs*. This is a consequence of the reward function considering *DS* in isolation. The agent therefore searches for graphs which optimally fit the *receptor* site without any consideration for other properties of those graphs. There are a few methods which could potentially address this. The first would be to hard-code filters, which prevent specific modifications that would lead to these unrealistic structures, into the environment. Alternatively, instead of rules which filter out certain actions that would lead to undesirable features, the agent could instead be presented with modifications which change whole functional groups. For example, instead of choosing only between adding a single carbon, nitrogen or oxygen atom, the agent's choices would also include the options to add carboxylic acid, aldehyde, amine or phenyl functional groups. In addition to these methods, *QED* and *SA* could be incorporated into the reward function. The goal would then be to maximize *QED* and *SA* simultaneously with *DS*. In addition to having high *DSs*, the resulting *ligands* would then also have high *QED* and *SA* scores as well, thus improving their utility.

The second shortcoming of the current implementation is the low accuracy of the *DS* calculations, as evidenced by the weakness of the correlation between *DS* and empirical *IC50* measurements for the set of reference *ligands*. One method for improving the simulation accuracy would be to use molecular dynamics simulations. These are simulations which, unlike the localized approximations used in this thesis, compute the forces acting on every atom in the system and apply Newtonian mechanics to calculate their corresponding accelerations, velocities and positions [Gelpi et al., 2015]. These simulations therefore model molecules as flexible bodies in motion, as opposed to static rigid bodies, and take solvent interactions into account. The goal here would be to improve the correlation between *DS* and *pIC50* measurements, as these determine a *ligand*'s potency in phenotypic whole-cell screenings.

Finally, there is the potential to avoid the inaccuracies and computational demands of simulations entirely by leveraging available *IC50* data. By first training a discriminative network to predict *IC50* values from molecular fingerprints, this predictive model could be used as the reward function for a generative agent. This could potentially be more useful than *DSs* calculated from simulations as *IC50* measurements are obtained from whole-cell phenotypic screenings, and thus they also account for off-target interactions within the cell. The reward returned from this predictive *IC50* model could of course also be combined with *QED* and *SA* as described above.

# Appendix A

## Experimental Parameters

### A.1 Network Structure

The structure of the behavioural and target ANNs, was the same as that of the original MolDQN network [Zhou et al., 2019]. The input to the networks was the 2048 bit circular ECFP\_4 fingerprint of state  $s_{t+1}$  concatenated with the steps remaining in the episode from state  $s_t$ . The structure of the network was a series of 5 dense linear layers composed of 2049 x 1024, 1024 x 512, 512 x 128, 128 x 32 and 32 x 1 parameter matrices respectively. The output neurons of each of these layers were passed through rectified linear unit activation functions, except for the final output neuron to which no activation function was applied.

The output was chosen to be a single neuron since it was impossible to define a fixed action space as the possible actions varied with each state of the environment. Thus, the Q-values for all states  $s_{t+1}$  accessible from state  $s_t$  were computed through separate forward passes.

In total the network contained 2,692,128 parameters and 1,696 hidden neurons. Both the behavioural and target ANNs shared the same structure.

The error criterion was the Huber loss (equation A.1)

$$l_n = \begin{cases} \frac{1}{2}(x_n - y_n)^2 & \text{if } |x_n - y_n| < \delta \\ \delta(|x_n - y_n| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \quad (\text{A.1})$$

with  $L = \text{mean}(l_n)$  and  $\delta = 1$ .

### A.2 Environment Parameters

The environment limited the maximum number of transitions per episode to 40, setting the terminal flag in the tuple returned to the agent at this point.

In the QED experiments the reward for a particular transitions was calculated using the RDKit QED package [RDKit, 2021a, Bickerton et al., 2012]. Given a molecular graph this returned a weighted sum of a number of chemical properties which are desirable in drug molecules.

In the ligand generation experiments the EA which implemented conformational search was run for 10 epochs per ligand candidate. The calculation of each conformer’s binding free energy during the course of an epoch was divided into 32 work units for GPU acceleration. The minimum binding free energy approximation from all conformations across all 10 epochs was returned to the agent as the molecule’s DS. These parameters are summarized in table A.1.

The centre of the docking box in which conformational search was run was located at coordinates ( $x = 6.98, y = 335.60, z = 8.06$ ) on the PfPI4K structure. The box contained 40 points at which

Table A.1: Environment and ANN Parameters

General Environment Parameters	
Molecule transitions per episode	40
Genetic algorithm runs per docking simulation	10
Number of parallel work units	32
Docking Box Parameters	
Centre	(6.98, 335.60, 8.06)
Number of Points	(40, 40, 40)
Spacing	0.375
DDQN ANN Hyperparameters	
Number of training episodes ( $N$ )	$5 \times 10^3$
Learning rate ( $\alpha$ )	$10^{-4}$
Discount factor ( $\gamma$ )	$9.99 \times 10^{-1}$
Backpropagation period	1 episode
Batch size ( $b$ )	40 transitions
Replay memory size ( $l$ )	$15 \times 10^3$ transitions
$\epsilon$ decay rate	$-9.9 \times 10^{-4}$ per episode
Target network soft-update fraction ( $\tau$ )	$10^{-3}$

a conformer’s binding free energy with the `receptor` was calculated in each direction, each spaced 0.375 Å apart. The parameters of the docking box are summarized in table A.1.

### A.3 Hyperparameters

The total number of training episodes, each generating 40 transitions of replay data, was limited to 5,000. The replay memory buffer  $l$  was limited to sliding window containing the most recent 15,000 transitions. The behaviour policy during training was  $\epsilon$ -greedy where  $\epsilon$  was annealed linearly from its initial value of 1.0 to 0.01 over the first 25% of training episodes and held at its final value thereafter (equation A.2)

$$\epsilon_t = \frac{\epsilon_f - \epsilon_i}{0.25 \times N} \times t + \epsilon_i \quad (\text{A.2})$$

with  $t$  corresponding to the current episode number, and  $\epsilon_i$  and  $\epsilon_f$  being the initial and final values of  $\epsilon$  respectively [Mnih et al., 2013, Mnih et al., 2015, Zhou et al., 2019]. The batch size for backpropagation  $b_i$  was maintained at 128 transitions throughout training. The target network parameters  $\phi$  were synchronized with the behaviour network parameters  $\theta$  on initialization, and updated using a soft-update (equation A.3)

$$\phi = \tau\theta + (1 - \tau)\phi \quad (\text{A.3})$$

whenever the behaviour network was updated [Lillicrap et al., 2015]. The target network thus received a fraction of the update applied to the behaviour network, where the fraction was defined by the soft-update parameter  $\tau = 0.001$ . These hyperparameters are summarized in table A.1.



## A.4 Discussion

The hyperparameters given in chapter 4 were identified through trial and error to avoid divergence within the constraints of the available hardware. For example, the replay memory was made to be as large as possible in order to prevent the agent from myopically overfitting its network to only the most recently generated transitions, but it had to be limited to 15,000 transitions due to limited available GPU memory. The total number of episodes was limited to 5,000 to limit the total experiment run-time. However, it is possible that the final **DSs** on which the agent converged would have been better if more training time was available. The batch size of 128 was chosen to be as small as possible while preventing the parameter updates from being too stochastic, and avoiding larger batches which would slow the network optimization method and increase the total run-time.

The parameters in tables A.1, specifically the number of points in the docking box and the number of genetic algorithm runs per docking simulation, were chosen to minimize the total training time. However, the consequence of this is that the **DSs** calculated with these parameters have limited accuracy. That is to say that many of the reference **ligands** from section 4.2.4 likely had better **DSs** than shown in figure 4.5. Similarly, many of the states obtained during the generation experiments likely also had better **DSs** however, these would not have been discovered due to the limited size of the docking box and number of genetic algorithm iterations. The algorithm would likely yield better results were these parameters adjusted to prioritize accuracy over time.

# Glossary

- a priori** Latin: “from what comes before” / “from first principles, before experience”. Refers to reasoning and deductions which follow from theory rather than empirical data. . 5
- activation** In the context of chemistry, activation refers to a reaction where a molecule transitions into a state in which it possesses the potential to undergo a specific chemical reaction. In the context of biochemistry this refers to biomolecules such as enzymes transitioning into states where they obtain the ability to perform their biological function. 2
- adenosine diphosphate** During biochemical processes ATP loses a phosphate group through hydrolysis thus forming ADP. 26
- adenosine triphosphate** An organic biomolecule which provides energy for biochemical processes. 26
- affinity** The strength of the bond formed between a ligand and receptor, usually quantified as either the change in Gibbs free energy during binding or the dissociation constant of the ligand-receptor complex. 2, 15, 26, 37
- agonist** An agonistic ligand. 2, 4
- agonistic** A type of bonding where the ligand activates a receptor upon bonding thus inducing a biological response. 2
- antagonist** An antagonistic ligand. 2, 4, 15
- antagonistic** A type of bonding where the ligand blocks the action of an agonist by bonding with a receptor, thus preventing its activation. 2, 33
- assay** An analytical laboratory procedure for determining the presence, concentration, or potency of an analyte. 3, 10
- bioassay** A type of assay which often involves the observation of an analyte’s interaction with a biological target *in vitro*. 3, 5, 22, 38
- charge distribution** The distribution of the electrons which comprise a molecule’s electron cloud, characterized by dense electronegative regions and sparse electropositive regions . 4
- chemotype** A particular class of molecules characterized by their common molecular scaffold. 4, 5, 26, 34, 39
- concentration** The ratio of solute in a solution to either solvent or total solution, expressed as mass per unit volume, molarity, parts per million, or a number of other measures . 3, 22, 38

**deactivation** In the context of chemistry this refers to a reaction through which a molecule transitions to state in which it exhibits a decreased propensity to undergo a certain reaction. In the case of biochemistry this refers to biomolecules such as enzymes being prevented from performing their biological functions. 2

**DNA** Deoxyribonucleic acid is a molecule composed of two polynucleotide chains that coil around each other to form a double helix carrying genetic instructions for the development, functioning, growth and reproduction of all known organisms and many viruses. 2

**electrostatic attraction** The attractive force defined by Coulomb's law occurring between oppositely charged nuclei, atoms or molecules. 2

**endogenous** Naturally occurring within the body. 2, 42, 45

**enzyme** Proteins which function as biological catalysts. 8, 15, 26, 27, 33, 34, 37, 38

**functional analogue** Molecules which share a structure that is similar enough that they can be expected to exhibit similar chemistry. 3, 4, 42

**half maximal inhibitory concentration** The concentration of a chemical species necessary to induce a 50% reduction in the biological activity of a target enzyme or protein. 34

**hydrogen bonding** A weak bond between two molecules resulting from an electrostatic attraction between a proton in one molecule and an electronegative atom in the other. 2, 16, 28, 29

**hydrolyzed** To be hydrolyzed is to undergo hydrolysis. That is, the severing of a chemical bond under the action of a water molecule. 26

**hydrophilic** An attraction to water. Polar molecules are highly soluble in polar solvents such as water due to their non-uniform charge distribution and are thus referred to as hydrophilic. Examples are short-chain alcohols like methanol and ethanol . 3

**hydrophobic** The propensity to repel water. Non-polar molecules, due to their uniform charge distribution, are insoluble in polar solvents like water, forming a boundary with the aqueous layer instead, and are thus referred to as hydrophobic. Examples are oils and fats . 3, 28

**hydrophobicity** See hydrophobic. 14

**in vitro** Latin: "in the glass". Refers to experiments conducted outside of a biological context, usually in some form of assay. 3, 15, 22, 38, 39, 43

**in vivo** Latin: "in the body". Refers to experiments conducted within a living organism. 3, 4, 10

**ligand** In the context of drug design, this is a molecule which binds to a receptor. Note that this is slightly different to the way the term is used in coordination chemistry, where it refers to a molecule which binds to a metal atom forming a coordination complex. 2, 4–7, 9, 10, 14–16, 22–30, 33–46, 48

**macromolecule** A molecule containing a very large number of atoms, such as a protein, nucleic acid, or synthetic polymer. 2, 5, 6, 36

**molecular polarizability** The response of a molecule's electron cloud to an externally-applied electric field . 3, 4

- molecular weight** The sum of the atomic weights of a molecule's atoms. 3, 4, 35
- pathology** The physiological manifestation of a deleterious biological process. 2, 4, 26
- pharmacodynamics** The biochemical and physiological effects of drugs on the body. 3
- pharmacokinetics** The manner in which a drug distributes throughout the body. 3
- photocatalyst** A material capable of absorbing light, thus imbuing it with kinetic energy which it can transfer to a reacting substance in order to facilitate a chemical reaction. 8
- physico-chemical** These are properties that are both physical and chemical in nature, and are therefore relevant to the physics of chemical interactions. Such as charge distribution, electronegativity, internal energy, entropy, enthalpy and so on. 3, 4, 6, 9, 10, 16, 19, 23–26, 40
- polarity** A separation of electric charge leading to a molecule or its chemical groups having an electric dipole moment, with a negatively charged end and a positively charged end . 3, 14
- protein** Large biomolecules and macromolecules that comprise one or more long chains of amino acid residues. 2, 3, 10, 15, 16, 26–28, 38
- quantitative estimate of drug-likeness** A quantitative score which combines a number of molecular properties in order to estimate the likelihood that a given chemical species would be well received by the body . 3
- receptor** The region within a macromolecule where another, smaller molecule will bind thus operating as a transducer of biological signals. 2–7, 9, 11, 14–16, 22, 26–30, 33, 34, 36–42, 44, 45, 47
- RNA** Ribonucleic acid is a polymeric molecule essential in various biological roles in coding, decoding, regulation and expression of genes. 2
- scaffold** In this context of chemistry this refers to the hydrocarbon backbone of a molecule excluding its substituents. 3–7, 14, 15, 25, 36–44
- selectivity** The degree to which a ligand binds at a given receptor relative to other receptors. Low selectivity results in off-target interactions and their concomitant physiological side-effects. 2–4, 10, 14
- solubility** The ability of a solid, liquid, or gaseous chemical substance (solute) to dissolve in solvent (usually a liquid) and form a solution. This depends on the solvent used, temperature and pressure, and is measured by the concentration of the saturated solution . 3, 4, 18, 19
- spectroscopy** The analysis of the presence, quantity and composition of an analyte in solution based wavelength and intensity of its absorptions or emissions when exposed to electromagnetic radiation. 3, 38
- synthetic accessibility** A quantitative measure of the complexity and cost of synthesizing a given chemical species. 3
- van der Waals interactions** The attractive or repulsive forces occurring between molecules due to their polarized electron clouds. 2

# Acronyms

- AAE** adversarial autoencoder. 22, 23
- ADP** adenosine diphosphate. 26
- AE** autoencoder. 5, 6, 8, 20, 23–25
- ANN** artificial neural network. 10, 17–20, 22, 27, 31–33, 46, 47
- ATP** adenosine triphosphate. 26, 33, 34, 36
- BA** binding affinity. 2–6, 10, 14–16, 25, 26, 29, 33, 36, 39, 42, 44
- CVAE** conditional variational autoencoder. 22
- DDQN** double deep Q-networks. 25, 30–35, 38, 40, 43, 47
- DS** docking score. 5–7, 16, 24, 26, 28–30, 33, 34, 36–46, 48
- EA** evolutionary algorithm. 4–6, 8, 10, 15, 23, 25, 42, 46
- EC** evolutionary computing. 15
- ECFP** extended-connectivity fingerprint. 11–13, 27, 46
- GAN** generative adversarial network. 5, 18, 19, 23–25
- GCPN** graph-convolutional policy network. 24
- IC50** half maximal inhibitory concentration. 34, 38, 39, 42–45
- JAK** Janus kinase. 22
- JTVAE** junction tree variational autoencoder. 24
- KL** Kullback-Leibler. 21
- LSTM** long short-term memory. 19
- MCTS** monte-carlo tree search. 16, 19
- MDP** Markov decision process. 17, 18
- ML** machine learning. 3–6, 8, 10, 24, 40

- ORGAN** objective-reinforced generative adversarial networks. 24
- ORGANIC** objective-reinforced generative adversarial network for inverse-design chemistry. 24
- PFPI4K** plasmodium falciparum phosphatidylinositol 4-kinase. 26, 27, 33, 34, 36, 39, 43, 46
- QED** quantitative estimate of drug-likeness. 3, 7, 18, 19, 23, 24, 26, 30, 34–36, 38, 40, 41, 43–46
- QSAR** quantitative structure-activity relationship. 8, 9, 14, 15
- RL** reinforcement learning. 4–6, 8, 10, 16–19, 23–27, 29, 30, 34, 38, 42–45
- RNN** recurrent neural network. 19
- SA** synthetic accessibility. 3, 18, 19, 26, 43–45
- SMILES** simplified molecular-input line-entry system. 16, 19–22
- VAE** variational autoencoder. 4, 16, 21, 22

# Bibliography

- [Andrychowicz et al., 2017] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2017). Hindsight Experience Replay. *arXiv*.
- [Atkins and de Paula, 2010] Atkins, P. and de Paula, J. (2010). *Physical Chemistry*. Oxford University Press, Oxford, 9 edition.
- [Aubert-Kato et al., 2017] Aubert-Kato, N., Fosseprez, C., Gines, G., Kawamata, I., Dinh, H., Cazenille, L., Estevez-Tores, A., Hagiya, M., Rondelez, Y.-N., and Bredeche, N. (2017). Evolutionary optimization of self-assembly in a swarm of bio-micro-robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 8, pages 59–66, New York, NY, USA. ACM.
- [Bickerton et al., 2012] Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. (2012). Quantifying the chemical beauty of drugs. *Nature Chemistry*, 4(2):90–98.
- [Blaschke et al., 2018] Blaschke, T., Olivecrona, M., Engkvist, O., Bajorath, J., and Chen, H. (2018). Application of Generative Autoencoder in De Novo Molecular Design. *Molecular Informatics*, 37(1-2):1700123.
- [Brown et al., 2004a] Brown, N., Mckay, B., and Gasteiger, J. (2004a). The de novo design of median molecules within a property range of interest. *Journal of Computer-Aided Molecular Design*, 18(12):761–771.
- [Brown et al., 2004b] Brown, N., Mckay, B., Gilardoni, F., and Gasteiger, J. (2004b). A Graph-Based Genetic Algorithm and Its Application to the Multiobjective Evolution of Median Molecules. *Journal of Chemical Information and Computer Sciences*, 44(3):1079–1087.
- [Bruice, 2011] Bruice, P. Y. (2011). *Organic Chemistry*. Pearson, sixth edition.
- [Burger et al., 2020] Burger, B., Maffettone, P. M., Gusev, V. V., Aitchison, C. M., Bai, Y., Wang, X., Li, X., Alston, B. M., Li, B., Clowes, R., Rankin, N., Harris, B., Sprick, R. S., and Cooper, A. I. (2020). A mobile robotic chemist. *Nature*, 583(7815):237–241.
- [Cavalcanti et al., 2008a] Cavalcanti, A., Shirinzadeh, B., Freitas, R. A., and Hogg, T. (2008a). Nanorobot architecture for medical target identification. *Nanotechnology*, 19(1).
- [Cavalcanti et al., 2008b] Cavalcanti, A., Shirinzadeh, B., Zhang, M., and Kretly, L. C. (2008b). Nanorobot hardware architecture for medical defense. *Sensors*, 8(5):2932–2958.
- [Chhabra, 2021] Chhabra, M. (2021). Biological therapeutic modalities. *Translational Biotechnology*, pages 137–164.

- [Coetzee and Nitschke, 2019] Coetzee, L. and Nitschke, G. (2019). Evolving optimal sun-shading building façades. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 393–394, New York, NY, USA. ACM.
- [Davis et al., 2021] Davis, A. P., Grondin, C. J., Johnson, R. J., Sciaky, D., Wieggers, J., Wieggers, T. C., and Mattingly, C. J. (2021). Comparative Toxicogenomics Database (CTD): update 2021. *Nucleic Acids Research*, 49.
- [DiMasi et al., 2016] DiMasi, J. A., Grabowski, H. G., and Hansen, R. W. (2016). Innovation in the pharmaceutical industry: New estimates of R&D costs. *Journal of Health Economics*, 47:20–33.
- [Dinh et al., 2015] Dinh, H. Q., Aubert, N., Noman, N., Fujii, T., Rondelez, Y., and Iba, H. (2015). An effective method for evolving reaction networks in synthetic biochemical systems. *IEEE Transactions on Evolutionary Computation*, 19(3):374–386.
- [Duim and Otto, 2017] Duim, H. and Otto, S. (2017). Towards open-ended evolution in self-replicating molecular systems. *Beilstein Journal of Organic Chemistry*, 13:1189–1203.
- [Eiben and Smith, 2015] Eiben, A. and Smith, J. (2015). *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2nd edition.
- [Elton et al., 2019] Elton, D. C., Boukouvalas, Z., Fuge, M. D., and Chung, P. W. (2019). Deep learning for molecular design - a review of the state of the art. *Molecular Systems Design and Engineering*, 4(4).
- [Fienberg et al., 2020] Fienberg, S., Eyermann, C. J., Arendse, L. B., Basarab, G. S., McPhail, J. A., Burke, J. E., and Chibale, K. (2020). Structural Basis for Inhibitor Potency and Selectivity of Plasmodium falciparum Phosphatidylinositol 4-Kinase Inhibitors. *ACS Infectious Diseases*, 6(11):3048–3063.
- [Gelpi et al., 2015] Gelpi, J., Hospital, A., Goñi, R., and Orozco, M. (2015). Molecular dynamics simulations: advances and applications. *Advances and Applications in Bioinformatics and Chemistry*, 8(1):37.
- [Glen and Payne, 1995] Glen, R. C. and Payne, A. W. R. (1995). A genetic algorithm for the automated generation of molecules within constraints. *Journal of Computer-Aided Molecular Design*, 9(2):181–202.
- [Gómez-Bombarelli et al., 2018] Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 4(2):268–276.
- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.
- [Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks.
- [Guimaraes et al., 2017] Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C., and Aspuru-Guzik, A. (2017). Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv preprint arXiv:1705.10843*.
- [Hagiya et al., 2016] Hagiya, M., Aubert-Kato, N., Wang, S., and Kobayashi, S. (2016). Molecular computers for molecular robots as hybrid systems. *Theoretical Computer Science*, 632:4–20.



- [Hagiya et al., 2014] Hagiya, M., Konagaya, A., Kobayashi, S., Saito, H., and Murata, S. (2014). Molecular Robots with Sensors and Intelligence. *Accounts of Chemical Research*, 47(6):1681–1690.
- [Harel and Radinsky, 2018] Harel, S. and Radinsky, K. (2018). Prototype-Based Compound Discovery Using Deep Generative Models. *Molecular Pharmaceutics*, 15(10):4406–4416.
- [He et al., 2021] He, Z., Tran, K. P., Thomassey, S., Zeng, X., Xu, J., and Yi, C. (2021). Multi-objective optimization of the textile manufacturing process using deep-Q-network based multi-agent reinforcement learning. *Journal of Manufacturing Systems*.
- [Huang et al., 2021] Huang, Y., Wu, D., Chen, M., Zhang, K., and Fang, D. (2021). Evolutionary optimization design of honeycomb metastructure with effective mechanical resistance and broadband microwave absorption. *Carbon*, 177:79–89.
- [Jang et al., 2020] Jang, S., Yoo, S., and Kang, N. (2020). Generative Design by Reinforcement Learning: Enhancing the Diversity of Topology Optimization Designs. *arXiv*.
- [Jensen, 2019] Jensen, J. H. (2019). A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. *Chemical Science*, 10(12):3567–3572.
- [Jin et al., 2018] Jin, W., Barzilay, R., and Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. *35th International Conference on Machine Learning, ICML 2018*, 5:3632–3648.
- [Jones et al., 1997] Jones, G., Willett, P., Glen, R. C., Leach, A. R., and Taylor, R. (1997). Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology*, 267(3):727–748.
- [Kadurin et al., 2017] Kadurin, A., Aliper, A., Kazennov, A., Mamoshina, P., Vanhaelen, Q., Khrabrov, K., and Zhavoronkov, A. (2017). The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, 8(7):10883–10890.
- [Khemchandani et al., 2020] Khemchandani, Y., O’Hagan, S., Samanta, S., Swainston, N., Roberts, T. J., Bollegala, D., and Kell, D. B. (2020). DeepGraphMolGen, a multi-objective, computational strategy for generating molecules with desirable properties: A graph convolution and reinforcement learning approach. *Journal of Cheminformatics*, 12(1).
- [Kim et al., 2021] Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B., Zaslavsky, L., Zhang, J., and Bolton, E. E. (2021). PubChem in 2021: new data content and improved web interfaces. *Nucleic Acids Research*, 49.
- [King et al., 2009] King, R. D., Rowland, J., Oliver, S. G., Young, M., Aubrey, W., Byrne, E., Liakata, M., Markham, M., Pir, P., Soldatova, L. N., Sparkes, A., Whelan, K. E., and Clare, A. (2009). The Automation of Science. *Science*, 324(5923):85–89.
- [Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv*.
- [Konda and Tsitsiklis, 2003] Konda, V. R. and Tsitsiklis, J. N. (2003). On Actor-Critic Algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166.
- [Lameijer et al., 2005] Lameijer, E. W., Bäck, T., Kok, J. N., and Ijzerman, A. P. (2005). Evolutionary algorithms in drug design. *Natural Computing*, 4(3):177–243.

- [Lillicrap et al., 2015] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv*.
- [Liu et al., 2020] Liu, B., Pappas, C. G., Ottel , J., Schaeffer, G., Jurissek, C., Pieters, P. F., Altay, M., Mari , I., Stuart, M. C. A., and Otto, S. (2020). Spontaneous Emergence of Self-Replicating Molecules Containing Nucleobases and Amino Acids. *Journal of the American Chemical Society*, 142(9):4184–4192.
- [Mendez et al., 2019] Mendez, D., Gaulton, A., Patr cia Bento, A., Chambers, J., De Veij, M., Paula Magari osMagari, M., Mosquera, J. F., Mutowo, P., Nowotka, M., Gordillo-Mar , M., Hunter, F., Junco, L., Mugumbate, G., Rodriguez-Lopez, M., Atkinson, F., Bosc, N., Radoux, C. J., Segura-Cabrera, A., Hersey, A., and Leach, A. R. (2019). ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research*, 47.
- [Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5.
- [Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.
- [Morris et al., 1639] Morris, G. M., Goodsell, D. S., Halliday, R. S., Huey, R., Hart, W. E., Bewley, R. K., and Olson, A. J. (1639). Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function. *Journal of Computational Chemistry*, 19(14):16391662.
- [Mysinger et al., 2012] Mysinger, M. M., Carchia, M., Irwin, J. J., and Shoichet, B. K. (2012). Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking. *J. Med. Chem.*, 55:23.
- [Ng et al., 1999] Ng, A. Y., Harada, D., and Russell, S. J. (1999). Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99*, pages 278–287, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [Nikolova and Jaworska, 2004] Nikolova, N. and Jaworska, J. (2004). Approaches to Measure Chemical Similarity - A Review. *QSAR and Combinatorial Science*, 22(9-10):1006–1026.
- [Ole Carstensen et al., 2011] Ole Carstensen, N., Dieterich, J. M., and Hartke, B. (2011). Design of optimally switchable molecules by genetic algorithms. *Phys. Chem. Chem. Phys.*, 13(7):2903–2910.
- [Polykovskiy et al., 2020] Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., Kadurin, A., Johansson, S., Chen, H., Nikolenko, S., Aspuru-Guzik, A., and Zhavoronkov, A. (2020). Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Frontiers in Pharmacology*, 11:1–17.
- [Polykovskiy et al., 2018] Polykovskiy, D., Zhebrak, A., Vetrov, D., Ivanenkov, Y., Aladinskiy, V., Mamoshina, P., Bozdaganyan, M., Aliper, A., Zhavoronkov, A., and Kadurin, A. (2018). Entangled Conditional Adversarial Autoencoder for de Novo Drug Discovery. *Molecular Pharmaceutics*, 15(10):4398–4405.
- [RDKit, 2021a] RDKit (2021a). rdkit.Chem.QED module — The RDKit 2021.03.1 documentation.

- [RDKit, 2021b] RDKit (2021b). The RDKit 2020.09.1 documentation.
- [Réda et al., 2020] Réda, C., Kaufmann, E., and Delahaye-Duriez, A. (2020). Machine learning applications in drug development. *Computational and Structural Biotechnology Journal*, 18:241–252.
- [Richard S. Sutton and Andrew G. Barto, 2018] Richard S. Sutton and Andrew G. Barto (2018). *Reinforcement learning: an introduction*. The MIT Press, Cambridge, MA, USA, second edition.
- [Roch et al., 2018] Roch, L. M., Häse, F., Kreisbeck, C., Tamayo-Mendoza, T., Yunker, L. P. E., Hein, J. E., and Aspuru-Guzik, A. (2018). ChemOS: Orchestrating autonomous experimentation. *Science Robotics*, 3(19):eaat5559.
- [Rogers and Hahn, 2010] Rogers, D. and Hahn, M. (2010). Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754.
- [Ruddigkeit et al., 2012] Ruddigkeit, L., van Deursen, R., Blum, L. C., and Reymond, J.-L. (2012). Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875.
- [Ruiz-Carmona et al., 2014] Ruiz-Carmona, S., Alvarez-Garcia, D., Foloppe, N., Garmendia-Doval, A. B., and Juhos, S. (2014). rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids. *PLoS Comput Biol*, 10(4):1003571.
- [Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- [Rupakheti et al., 2015] Rupakheti, C., Virshup, A., Yang, W., and Beratan, D. N. (2015). Strategy to discover diverse optimal molecules in the small molecule universe. *Journal of Chemical Information and Modeling*, 55(3):529–537.
- [Sanchez-Lengeling and Aspuru-Guzik, 2018] Sanchez-Lengeling, B. and Aspuru-Guzik, A. (2018). Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365.
- [Sanchez-Lengeling et al., 2017] Sanchez-Lengeling, B., Outeiral, C., Guimaraes, G. L., and Aspuru-Guzik, A. (2017). Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). *ChemRxiv*.
- [Santos-Martins et al., 2021] Santos-Martins, D., Solis-Vasquez, L., Tillack, A. F., Sanner, M. F., Koch, A., and Forli, S. (2021). Accelerating AutoDock 4 with GPUs and Gradient-Based Local Search. *Journal of Chemical Theory and Computation*, 17(2):1060–1073.
- [Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv*.
- [Sheridan et al., 2000] Sheridan, R. P., SanFeliciano, S. G., and Kearsley, S. K. (2000). Designing targeted libraries with genetic algorithms. *Journal of Molecular Graphics and Modelling*, 18(4-5):320–334.
- [Sumita et al., 2018] Sumita, M., Yang, X., Ishihara, S., Tamura, R., and Tsuda, K. (2018). Hunting for Organic Molecules with Artificial Intelligence: Molecules Optimized for Desired Excitation Energies. *ACS Central Science*, 4(9):1126–1133.

- [Supady et al., 2015] Supady, A., Blum, V., and Baldauf, C. (2015). First-Principles Molecular Structure Search with a Genetic Algorithm. *Journal of Chemical Information and Modeling*, 55(11):2338–2348.
- [Trott and Olson, 2010] Trott, O. and Olson, A. J. (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):455–461.
- [Wang et al., 2019] Wang, Y., Zhang, S., Li, F., Zhou, Y., Zhang, Y., Wang, Z., Zhang, R., Zhu, J., Ren, Y., Tan, Y., Qin, C., Li, Y., Li, X., Chen, Y., and Zhu, F. (2019). Therapeutic target database 2020: enriched resource for facilitating research and early development of targeted therapeutics. *Nucleic Acids Research*, 48:1031–1041.
- [Wikipedia, 2021] Wikipedia (2021). Cyclohexane Conformation.
- [Wishart et al., 2018] Wishart, D. S., Feunang, Y. D., Guo, A. C., Lo, E. J., Marcu, A., Grant, J. R., Sajed, T., Johnson, D., Li, C., Sayeeda, Z., Assempour, N., Iynkkaran, I., Liu, Y., Maciejewski, A., Gale, N., Wilson, A., Chin, L., Cummings, R., Le, D., Pon, A., Knox, C., and Wilson, M. (2018). DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46.
- [Yoshikawa et al., 2018] Yoshikawa, N., Terayama, K., Sumita, M., Homma, T., Oono, K., and Tsuda, K. (2018). Population-based De Novo Molecule Generation, Using Grammatical Evolution. *Chemistry Letters*, 47(11):1431–1434.
- [You et al., 2018] You, J., Liu, B., Ying, R., Pande, V., and Leskovec, J. (2018). Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. *Advances in Neural Information Processing Systems*, 2018-Decem(NeurIPS):6410–6421.
- [Yu et al., 2016] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2016). SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *CoRR*, abs/1609.0.
- [Zhou et al., 2019] Zhou, Z., Kearnes, S., Li, L., Zare, R. N., and Riley, P. (2019). Optimization of Molecules via Deep Reinforcement Learning. *Scientific Reports*, 9(1):10752.