UNIVERSITY OF OXFORD



Novel Reaction-Based *De Novo* Molecular Design in 2D & 3D using Deep Q-Learning

---

An Goto

Balliol College,
EPSRC & MRC Systems Approaches to Biomedical Science CDT,
Department of Statistics

A thesis submitted in partial fulfilment of the requirements of the degree of Doctor of Philosophy

Michaelmas 2022

Abstract

There has been a growth of interest in *de novo* molecular design methods in recent years, thanks to exciting new developments in artificial intelligence, and deep learning in particular. Molecule Deep Q-Networks (MolDQN) is a *de novo* algorithm developed by Zhou *et al.* in 2019 that combines simple chemical actions – adding or removing atoms or bonds – together with reinforcement learning. Consequently, MolDQN enables "local" search of chemical space by enabling smaller chemical modifications. However, often MolDQN's atom/bond-based actions result in chemically infeasible or synthetically inaccessible molecules. To address this shortcoming, I developed new reinforcement learning-based *de novo* methods as described in Chapter 2: Reaction Deep Q-Networks (RxnDQN) and Reaction-Molecule Deep Q-Networks (RxnMolDQN). These methods introduced new action spaces that are based on available reagents and the most widely used, high-yield reactions used in medicinal chemistry. Therefore, my new action spaces enable "global" search of chemical space by performing larger chemical modifications.

In contrast to the atom- and bond-based action space of MolDQN, the action space of RxnDQN incorporates the top ten reactions used in medicinal chemistry as reported by Roughley *et al.* in 2011. I hypothesized that this would enable more synthetically accessible molecules to be generated. I also tested whether hybridizing these two action spaces in RxnMolDQN would enhance the exploration of chemical space by enabling both "local" and "global" modifications in chemical space.

In Chapter 3, I also explored novel structure-based variants by modifying the reward function to accommodate protein-ligand interactions in a protein binding site of interest. This was achieved

by incorporating protein-ligand docking into the DQN methods (MolDQN, RxnDQN, and RxnMolDQN), in my new non-constrained structure-based *de novo* methods (*i.e.*, nc3D-DQN methods):

- non-constrained three-dimensional Molecule Deep Q-Networks (nc3D-MolDQN)
- non-constrained three-dimensional Reaction Deep Q-Networks (nc3D-RxnDQN)
- non-constrained three-dimensional Reaction-Molecule Deep Q-Networks (nc3D-RxnMolDQN)

I further extended these new non-constrained structure-based *de novo* methods by incorporating constrained docking (*i.e.*, c3D-DQN methods) in Chapter 4: c3D-MolDQN, c3D-RxnDQN, and c3D-RxnMolDQN where "c" implies constrained, which allow for *de novo* covalent ligand design, as well as opening up the potential for fragment-based *de novo* design.

The 2D-DQN methods (*i.e.*, MolDQN, RxnDQN, and RxnMolDQN) were tested using 100 clinically approved drugs from DrugBank. The Malhotra set is a 297 collection of substructurally related ligand-pairs solved in complex with the same protein partner, which was used in this thesis to design the larger ligand from the smaller ligand. The 3D-DQN methods (*i.e.*, nc3D-DQN and c3D-DQN methods) were tested using 51 ligand-pairs from the Malhotra set and 17 covalent ligands for SARS-CoV-2 main protease (M^{pro}) from Fragalysis, respectively. Fragalysis is a cloud-based platform that has reported X-ray crystal structures of SARS-CoV-2 M^{pro}. I also introduced a new pharmacophoric-based reward function using normalized SuCOS (2019) in the nc3D-DQN methods, which was tested using 45 ligand-pairs from the Malhotra set.

The 2D-DQN methods were found to be better at exploring chemical space than 3D-DQN methods. However, 3D-DQN methods were found to be more suitable if one wants to obtain molecules that are drug-like but also capable of binding in a protein's active site. This is achieved by imposing constraints on the chemical space exploration by introducing 3D-based reward functions such as predicted ligand efficiency obtained by docking. Normalized SuCOS as a reward prove to be more difficult to optimize for than predicted ligand efficiency. Overall, however, our reaction-based action spaces when applied to both 2D-DQN and 3D-DQN methods enabled the generation of new molecules that are more synthetically accessible while simultaneously improving chemical space exploration and other molecular properties important to drug discovery. These new algorithms provide a framework to perform global search of synthetically accessible chemical space whilst optimizing for multiple molecular properties vital to real-world drug discovery. My structure-based *de novo* methods have proven to be a promising way to propose novel inhibitors of SARS-CoV-2 main protease. The reaction variants of RxnDQN (both 2D and 3D-DQN methods) also define the reagents and reactions needed to synthesize a *de novo* designed compound, so will also prove to be useful in guiding synthesis and virtual library design.

Acknowledgements

Thank you to my supervisors, Prof. Garrett M. Morris and Prof. Charlotte Deane, for their constant support with my work since the rotation project. In particular to Garrett, who has assisted me wholeheartedly during unforeseen circumstances and helped me to move forward with this project throughout my DPhil degree. Thank you for providing me with the opportunity to conduct such a rewarding project.

Also thank you to my industrial supervisors, Torsten Schindler, Kristian Birchall, Will Pitt, Jiye Shi, Andy Merritt, and Lewis Vidler, for their valuable inputs throughout the project. I enjoyed the industrial visits that I made during the first year of my DPhil.

I would also like to thank the members of Oxford Protein Informatic Group (OPIG) for making my time at Oxford memorable. Special thanks to Susan for teaching me how to use RDKit from scratch, Fergus I and Fergus II for helping me with code problems, and Marc for assisting me with the covalent docking pipeline.

And lastly, special acknowledgement to my family, who has constantly supported and truly believed in my potential. I am almost always far away from home but thank you for always being there for me.

Table of Contents

1 Introduction 1

1.1 Overview 1

1.2 Drug Discovery 1

1.3 *De novo* Drug Design 12

1.4 Ligand-Based *De Novo* Drug Design 15

1.5 Structure-Based *De Novo* Drug Design 17

1.6 Machine Learning and AI 21

1.7 *De Novo* Drug Design using AI 24

1.8 Evaluation of *De Novo* Drug Design 36

1.9 Future Directions in *De Novo* Drug Design 37

1.10 Project Scope 38

2 Exploration of Novel Action Spaces in Deep Q-Networks 42

2.1 Overview 42

2.2 Background 42

2.2.1 Markov Decision Process (MDP) 42

2.2.2 Reinforcement Learning (RL) 46

2.2.3 Deep Q-Networks (DQN) 49

2.2.4 Deep Q-Learning 51

2.2.5 Action Space 54

2.2.6 Multi-objective Optimization in Reinforcement Learning 54

2.2.7 RDKit 55

2.2.8 TensorFlow 56

2.2.9 Reaction Cheminformatics 56

2.2.10 Penalized logP & Synthetic Accessibility Score (SAscore) 58

2.2.11 Quantitative Estimate of Druglikeness (QED) 61

2.2.12 Reaxys Database 63

2.2.13 DrugBank Database 63

2.2.14 Project Aims 64

2.3 Methodology 65

2.3.1 Molecule Deep Q-Networks (MolDQN) 65
2.3.2 Reaction Deep Q-Networks (RxnDQN) 72
2.3.3 Hybridized Action Spaces in Deep Q-Networks (RxnMolDQN) 75
2.3.4 Modifications to MolDQN 76
3 Exploration of Docking Methods in Deep Q-Networks 78
3.1 Overview 78
3.2 Background 78
3.2.1 Introduction to Docking 78
3.2.2 Machine Learning in Docking 80
3.2.3 AutoDock Vina 82
3.2.4 Open Babel 86
3.2.5 Predicted Ligand Efficiency (LE) 87
3.2.6 Malhotra Dataset 87
3.2.7 Normalized SuCOS 88
3.2.8 Project Aim 91
3.3 Methodology 92
3.3.1 Non-Constrained Docking in Deep Q-Networks (nc3D-DQN) 92
3.3.2 Comparison to 2D-DQN Methods 95
3.3.3 Multi-objective Optimization in nc3D-DQN Methods 95
4 Constrained Docking in Deep Q-Networks: SARS-CoV-2 Main Protease 97
4.1 Overview 97
4.2 Background 97
4.2.1 Constrained Docking Versus Covalent Docking 97
4.2.2 SARS-CoV-2 Main Protease 98
4.2.3 M^{pro} Dataset from XChem's Fragalysis 101
4.2.4 Project Aim 101
4.3 Methodology 102
4.3.1 Constrained Docking in Deep Q-Networks (c3D-DQN) 102
4.3.2 Comparison to nc3D-DQN Methods 106
5 Methods of Evaluation for 2D-DQN and 3D-DQN 107
5.1 Overview 107
5.2 Dataset for Method Validations 107

5.3 Exploration of Hyperparameters 110
5.4 Computational Details for Method Validations 113
5.4.1 Reagents in Reaction-based Action Space 113
5.4.2 Hyperparameters (JSON) 114
5.4.3 Molecular Weight Filter 115
5.4.4 AutoDock Vina Details 116
5.5 Methods of Assessment 118
5.5.1 Validity 118
5.5.2 Synthetic Accessibility 118
5.5.3 Reward 119
5.5.4 Walters' Compound Quality 119
5.5.5 Chemical Space Coverage 120
5.5.6 Silly Walks 122
5.5.7 Rediscovery 122
5.5.8 Optimization Efficiency 123
5.5.9 Uniqueness 123
5.5.10 Computational Efficiency 124
5.5.11 Tanimoto Similarity 124
5.5.12 Normalized SuCOS Distributions 125
5.5.13 Lipinski's Rule of Five 126
5.5.14 Multi-objective Optimizations 126
5.5.15 Comparison against Published Methods 127
6 Results and Discussion 128
6.1 Overview 128
6.2 Validity 128
6.3 Synthetic Accessibility 129
6.4 Reward (QED) 131
6.5 Reward (Penalized logP) 132
6.6 Reward (LE) 133
6.6.1 nc3D-DQN Methods 133
6.6.2 c3D-DQN Methods 136
6.7 Reward (Normalized SuCOS) 138
6.8 Walters' Compound Quality 139

6.9 Rediscovery 141
6.9.1 2D-DQN Methods 141
6.9.2 nc3D-DQN (LE) Methods 142
6.9.3 nc3D-DQN (nSuCOS) Methods 143
6.9.4 c3D-DQN Methods 144
6.10 Tanimoto Similarity 145
6.10.1 Representative Examples for Top 10 Tanimoto Similarity 145
6.10.2 Tanimoto Similarity Results for DQN Methods 149
6.11 Chemical Space Coverage 158
6.12 Normalized SuCOS Distribution 163
6.13 Silly Walks 168
6.14 Optimization Efficiency 170
6.14.1 2D-DQN Methods 170
6.14.2 nc3D-DQN (LE) Methods 171
6.14.3 nc3D-DQN (nSuCOS) Methods 172
6.14.4 c3D-DQN Methods 174
6.15 Uniqueness 175
6.16 Computational Efficiency 177
6.17 Lipinski's Rule of Five 178
6.18 Multi-objective Optimization 180
6.18.1 2D-DQN Methods 180
6.18.2 nc3D-DQN (LE) Methods 183
6.18.3 nc3D-DQN (nSuCOS) Methods 184
6.18.4 c3D-DQN Methods 186
6.19 Comparison of 2D-DQN against Other Published Methods 187
6.20 Conclusion and Future Directions 188
7 Conclusion and Future Directions 195
References 198
Appendix A: Reaction SMARTS 220
Appendix B: Reagents and Starting Molecules 224
Appendix C: Chemical Space Coverage 240

Appendix D: Validity 242
Appendix E: Synthetic Accessibility 246
Appendix F: Reward 249
Appendix G: Walters' Compound Quality 284
Appendix H: Silly Walks 288
Appendix I: Rediscovery 291
Appendix J: Optimization Efficiency 295
Appendix K: Uniqueness 307
Appendix L: Computational Efficiency 311
Appendix M: Tanimoto Similarity 315
Appendix N: Sample Workflow for c3D-DQN Methods 338

List of Figures

Figure 1.1: Structure of morphine…….....1
Figure 1.2: History of important events in drug discovery…….2
Figure 1.3: Structures of (a) penicillin and (b) cephalosporin…….....3
Figure 1.4: Structures of (a) Ivermectin and (b) lovastatin…….……...4
Figure 1.5: Structures of (a) maraviroc, (b) raltegravir, and (c) saquinavir……...6
Figure 1.6: Structure of temsirolimus……7
Figure 1.7: The pipeline in drug discovery……....9
Figure 1.8: Schematic representation of *de novo* drug design methods……...13
Figure 1.9: Development of AurA inhibitors……...16
Figure 1.10: Categories of scoring functions…….…..18
Figure 1.11: Classification of physics-based scoring functions……..19
Figure 1.12: Schematic of relationships between Artificial Intelligence, Machine Learning, Neural Networks, Deep Reinforcement Learning, Deep Learning, and Reinforcement Learning……....21
Figure 1.13: Schematic representations of the neural network architectures for *de novo* drug design……25
Figure 1.14: Architectures of (a) REINVENT and (b) GraphNet……....26
Figure 1.15: Architecture of ChemVAE……..30
Figure 1.16: Architecture of ORGAN…….….33
Figure 1.17: Scope of the Deep Q-Networks methods developed during the project…….38
Figure 2.1: A simple example of a Markov Decision Process (MDP)……45
Figure 2.2: Architecture of DQN…….…50
Figure 2.3: Example of most common fragments present in the million PubChem molecules…59
Figure 2.4: Distribution of Synthetic Accessibility Score……....60
Figure 2.5: The parameters in MDP…….…66
Figure 2.6: Flowchart for MolDQN, RxnDQN, and RxnMolDQN…….……70
Figure 2.7: Schematics of the top 10 synthetic groups used in medicinal chemistry…….….72
Figure 2.8: Four types of amide-related reactions in the new reaction action space……...74
Figure 3.1: An example of smaller and larger ligand-pair from the Malhotra set……...88
Figure 3.2: Schematics of normalized SuCOS score……89
Figure 3.3: Flowchart for nc3D-DQN (LE) Methods……..93
Figure 3.4: Flowchart for nc3D-DQN (nSuCOS) Methods…….…94
Figure 4.1: Schematic of two potential types of constrained docking…….98

Figure 4.2: (a) The catalyzed reaction of M^{pro}. (b) The binding site of SARS-CoV-2 main protease…….100
Figure 4.3: Covalent attachment point for new covalent inhibitors…..……101
Figure 4.4: Flowchart for c3D-DQN methods……...103
Figure 4.5: The newly generated molecule from DQN in step **(4)** shown in (a) and the compound-amino acid adduct after reacting with cysteine residue shown in (b)…….104
Figure 4.6: (a) The docked pose after performing constrained docking between flexible residue, and prepared protein PDBQT file in step **(8)**. (b) The docked pose from the PDBQT file for score_only docking using AutoDock Vina to obtain the final docking score in step **(9)**……105
Figure 5.1: Workflow to select 100 chemically diverse drugs from DrugBank……....108
Figure 5.2: Workflow to select 51 ligand-pairs and 45 ligand-pairs from the Malhotra set…..109
Figure 5.3: Workflow to select 17 covalent ligands from the Fragalysis database of SARS-CoV-2 main protease crystal structures……...109
Figure 5.4: Procedure to compute relative chemical space coverage using UMAP……..121
Figure 6.1: Average synthetic accessibility for 2D-DQN and 3D-DQN methods……..…..130
Figure 6.2: Top 10 predicted ligand efficiency scored compounds for nc3D-DQN methods…135
Figure 6.3: Top 10 predicted ligand efficiency scored compounds for c3D-DQN methods.....137
Figure 6.4: DrugBank drug selected to show representative examples of Tanimoto similarity for 2D-DQN methods……...145
Figure 6.5: Malhotra smaller ligand selected to show representative top 10 Tanimoto similarity for nc3D-DQN (LE) methods together with its crystal structure and molecular surface…..….146
Figure 6.6: Malhotra smaller ligand selected to show representative top 10 Tanimoto similarity for nc3D-DQN (nSuCOS) together with its crystal structure and molecular surface……....147
Figure 6.7: Covalent ligand selected to show representative top 10 Tanimoto similarity for c3D-DQN methods together with its crystal structure and molecular surface…….……..148
Figure 6.8: Top 10 Tanimoto similarity compounds for 2D-DQN methods…….…150
Figure 6.9: Top 10 Tanimoto similarity compounds for nc3D-DQN (LE) methods…….…152
Figure 6.10: Representative 3D docked pose for nc3D-DQN (LE)..…….……153
Figure 6.11: Representative 3D docked pose for nc3D-DQN (nSuCOS)…….….155
Figure 6.12: Top 10 Tanimoto similarity compounds for nc3D-DQN (nSuCOS) methods…..156
Figure 6.13: Top 10 Tanimoto similarity compounds for c3D-DQN methods…….…157
Figure 6.14: Representative 3D docked pose for c3D-DQN……..……...158
Figure 6.15: Per-drug relative percentage chemical space coverage for 2D-DQN methods….159
Figure 6.16: Five DrugBank compounds where per-drug percentage chemical space coverage computed using UMAP scored poorly for RxnDQN……..160
Figure 6.17: Per-drug relative percentage chemical space coverage for nc3D-DQN (LE)……161

Figure 6.18: Per-drug relative percentage chemical space coverage for nc3D-DQN (nSuCOS) ……...162
Figure 6.19: Per-drug relative percentage chemical space coverage for c3D-DQN……..163
Figure 6.20: Top 10 normalized SuCOS compounds for nc3D-DQN (LE) methods ……...164
Figure 6.21: Top 10 normalized SuCOS compounds for c3D-DQN methods …….166
Figure 6.22: Average silly score for 2D-DQN and 3D-DQN methods……..……...169
Figure 6.23: Reward progression for 2D-DQN methods……...170
Figure 6.24: Reward progression for nc3D-DQN (LE) methods……..172
Figure 6.25: Reward progression for nc3D-DQN (nSuCOS) methods…….173
Figure 6.26: Reward progression for c3D-DQN methods……..…174
Figure 6.27: Distribution of QED values when performing multi-objective optimization between QED and Tanimoto similarity for 2D-DQN methods……....181
Figure 6.28: Distribution of Tanimoto similarity when performing multi-objective optimization between QED and Tanimoto similarity for 2D-DQN methods…….…….182
Figure 6.29: Reward progression for nc3D-DQN (LE) methods in multi-objective optimization between LE and Tanimoto similarity…….…….....184
Figure 6.30: Reward progression for nc3D-DQN (nSuCOS) methods in multi-objective optimization between nSuCOS and Tanimoto similarity…….……..185
Figure 6.31: Reward progression for c3D-DQN methods in multi-objective optimization between LE and Tanimoto similarity…….…….....186

List of Tables

Table 2.1: Summary of *de novo* methods developed for the 2D-DQN methods…….65
Table 3.1: Comparison of features for AutoDock family and GOLD…….79
Table 3.2: Summary of *de novo* methods developed for the nc3D-DQN methods……..…91
Table 4.1: The eleven cleavage sites for SARS-CoV-2 main protease……...99
Table 4.2: Summary of *de novo* methods developed for constrained/covalent c3D-DQN…...102
Table 5.1: Metrics used to assess 2D-DQN and 3D-DQN methods in Chapter 6…….107
Table 5.2: List of 2D-DQN methods……..…111
Table 5.3: List of nc3D-DQN methods……..…112
Table 5.4: List of c3D-DQN methods……....113
Table 6.1: QED for 2D-DQN methods……..132
Table 6.2: Penalized logP for 2D-DQN methods……..133
Table 6.3: Predicted ligand efficiency for nc3D-DQN methods ……..134
Table 6.4: Predicted ligand efficiency for c3D-DQN methods…….136
Table 6.5: Normalized SuCOS for nc3D-DQN methods……..139
Table 6.6: Average Walters' compound quality for 2D-DQN and 3D-DQN methods ……140
Table 6.7: Rediscovery for 2D-DQN methods…….……141
Table 6.8: Rediscovery of Malhotra smaller ligands for nc3D-DQN (LE) methods…….142
Table 6.9: Rediscovery of Malhotra larger ligands for nc3D-DQN (LE) methods……..…….142
Table 6.10: Rediscovery of Malhotra smaller ligands for nc3D-DQN (nSuCOS) methods…..143
Table 6.11: Rediscovery of Malhotra larger ligands for nc3D-DQN (nSuCOS) methods……143
Table 6.12: Rediscovery of c3D-DQN methods……....145
Table 6.13: Average Tanimoto similarity for 2D-DQN methods…….……149
Table 6.14: Average Tanimoto similarity for nc3D-DQN (LE) methods……..151
Table 6.15: Average Tanimoto similarity for nc3D-DQN (nSuCOS) methods…….154
Table 6.16: Average Tanimoto similarity for c3D-DQN methods……156
Table 6.17: Percentage of generated compounds that satisfy normalized SuCOS threshold for nc3D-DQN (LE) methods…….……163
Table 6.18: Percentage of generated compounds that satisfy normalized SuCOS threshold for nc3D-DQN (nSuCOS) methods…….……165
Table 6.19: Percentage of generated compounds that satisfy normalized SuCOS threshold for c3D-DQN methods…….……167
Table 6.20: Average percentage uniqueness of the molecules generated by the first 1,500 episodes of 2D-DQN methods……...…175

Table 6.21: Average percentage uniqueness of the molecules generated by the first 150 episodes of 3D-DQN methods……...176
Table 6.22: Computational efficiency for 2D-DQN and 3D-DQN methods…….177
Table 6.23: Lipinski's rule of five for 3D-DQN methods……..178
Table 6.24: Lipinski's rule of five for 2D-DQN methods using the same dataset as 3D-DQN methods……...179
Table 6.25: Comparison of QED and validity for best models in 2D-DQN methods against literature methods using the reward function, QED……...188
Table 6.26: Comparison of penalized logP and validity for best models in 2D-DQN methods against literature methods using the reward function, penalized logP……...188

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
3D-CoMFA	3D Molecular Field Scoring Model of Comparative Molecular Field Analysis
GlcCerase	Acid β-glucosidase
AAE	Adversarial Autoencoder
ACH	Acetylcholine (ligand from PDB ID: 2ha4)
AD4	AutoDock4
ADMET	Adsorption, Distribution, Metabolism, Excretion, and Toxicity
ADS	Asymmetric Double Sigmoidal
ADT	AutoDock Tools
AE	Autoencoder
AI	Artificial Intelligence
Akt	Protein Kinase B
AL	Adversarial Learning
AlogP	Octanol-Water Partition Coefficient
API	Application Programming Interface
ARC	Advanced Research Computing (the University of Oxford's supercomputer)
AurA	Aurora A kinase
BO	Bayesian Optimization
BSD	Berkeley Source Distribution
c3D	Constrained Three-Dimensional
CAESA	Computer Assisted Estimation of Synthetic Accessibility

ChEMBL	Database of Bioactive Drug-like Small Molecules from the European Molecular Biology Laboratory
ChemVAE	Chemical Variational Autoencoder
CNN	Convolutional Neural Networks
COS	Combined Overlap Score
CONCERTS	Creation of Novel Compounds by Evaluation of Residues at Target Sites
COVID-19	Coronavirus Disease 2019
CPU	Central Processing Unit
Cys	Cysteine
DNA	Deoxyribose Nucleic Acid
DOGS	Design of Genuine Structures
DQN	Deep Q-Network(s)
ECFP	Extended-Connectivity Fingerprints
EG	Epsilon Greedy
EMA	Exponential Moving Average
EMEA	European Medicines Evaluation Agency
Enc-Dec	Encoder-Decoders
FB1	2-Chlorobenzenesulfonamide (ligand from PDB ID: 2weh)
FB2	Benzenesulfonamide (ligand from PDB ID: 2wej)
FDA	Food and Drug Administration
GA	Genetic Algorithm
GAN	Generative Adversarial Network
GCPN	Graph Convolutional Policy Network
GP	Gaussian Process

GPCR	G-Protein Coupled Receptor
GRU	Gated Recurrent Unit
HBA	Hydrogen-Bond Acceptors
HBD	Hydrogen-Bond Donors
hCA II	Human Carbonic Anhydrase II
His	Histidine
HIV	Human Immunodeficiency Virus
InChI	International Chemical Identifier
IUPAC	International Union of Pure & Applied Chemistry
JT-VAE	Junction Tree Variational Autoencoder
KL	Kullback-Leibler
LE	Predicted Ligand Efficiency
LGA	Lamarckian Genetic Algorithm
logP	Logarithmic Partition Coefficient
LRo5	Lipinski's Rule of Five
MDP	Markov Decision Process
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MolDQN	Molecule Deep Q-Networks
M^{pro}	Main Protease or SARS-CoV-2 Main Protease
nc3D	Non-constrained Three-Dimensional
NMEs	New Molecular Entities
NMR	Nuclear Magnetic Resonance
nSuCOS	Normalized SuCOS

PDB	Protein Data Bank
pEC_{50}	Half Maximal Effective Concentration
pIC_{50}	Half Maximal Inhibitory Concentration
PLIGs	Protein-Ligand Interaction Graphs
PLIF	Protein Ligand Interaction Fingerprint
p.logP	Penalized logP
PL^{pro}	Papain-Like Protease
ORGAN	Objective Reinforced Generative Adversarial Networks
Q	Quality
QED	Quantitative Estimate of Druglikeness
Q-Learning	Quality-Learning
QSAR	Quantitative Structure-Activity Relationship
ReLU	Rectified Linear Units
RGB	Red-Green-Blue
RL	Reinforcement Learning
RW	Random Walk
RMSD	Root Mean Square Deviation
RNN(s)	Recurrent Neural Network(s)
RxnDQN	Reaction Deep Q-Networks
RxnMolDQN	Reaction-Molecule Deep Q-Networks
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus 2
SARSA	State Action Reward State Action
SAscore	Synthetic Accessibility Score
SCK	Succinyldicholine (ligand from PDB ID: 2ha2)

SELFIES	Self-referencing Embedded Strings
SIM	Similarity Score
SMARTS	SMILES Arbitrary Target Specification
SMILES	Simplified Molecular-Input Line-Entry System
T	Tanimoto Similarity
TD	Temporal Difference
TOPAS	TOPology-Assigning System
t-SNE	T-distributed Stochastic Neighbor Embedding
UMAP	Uniform Manifold Approximation and Projection
UniProt	Universal Protein Resource
VAE	Variational Autoencoder
ZINC	ZINC is not Commercial

1 Introduction

1.1 Overview

In this chapter, I introduce drug discovery, its history, pipeline, and challenges involved (Section 1.2). Computers have a significant impact in this field, and one of these areas is the automated design of new molecules, or "*de novo*" design. Various types of *de novo* drug design methods are discussed (Section 1.3) and later I focus on how ligand-based and structure-based *de novo* drug design methods work (Sections 1.4-1.5). After introducing artificial intelligence (AI) (Section 1.6), I explain how AI has been incorporated into *de novo* drug design recently (Section 1.7) and how these methods are evaluated (Section 1.8). Lastly, I describe the future directions of the field by highlighting its current challenges (Section 1.9) and the project scope of this work (Section 1.10).

1.2 Drug Discovery

The advancement of modern drug discovery consists of three main periods (Figure 1.2). The first period arose during the 19th century when drug discovery was advanced by medicinal chemists, who typically extracted compounds from plants that were known to have therapeutic effects (Newman *et al.*, 2000). Morphine (Figure 1.1) is one of the examples, which was isolated from opium extracts in 1815 by Sertuner (Ballester and Mitchell, 2010).

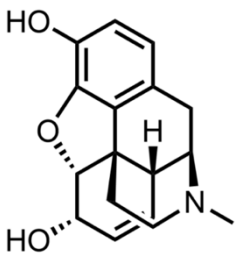


Figure 1.1: Structure of morphine isolated in 1815 by Sertuner.

Nineteenth-century medicine was advanced by the emergence of analytical chemistry in the late 19th century when individual bioactive ingredients were more readily isolated from plants. However, the purity levels were far from modern pharmaceutical standards (G. Thomas, 2007). Some of the essential foundations of chemical theory including Avogadro's atomic hypothesis and Mendeleev's periodic table were established by 1870 (Moore, 1918).

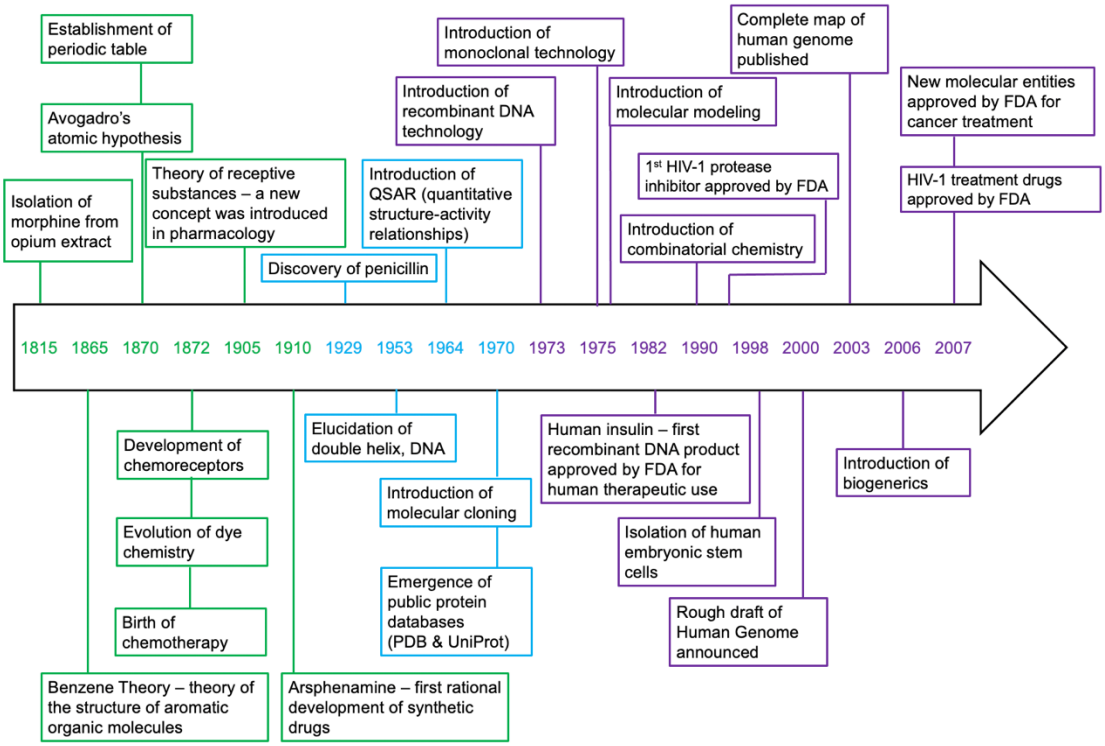


Figure 1.2: History of important events in drug discovery where the timescale of three main periods is shown in green (first period), blue (second period), and purple (last period).

Benzene theory (concept of aromaticity) by August Kekule in 1865 (Rocke, 2015) led to the evolution of dye chemistry that profoundly influenced medicine (Travis, 2019). This theory suggested the selective affinity of dyes for biological tissues, which postulated the theory of chemoreceptors by Paul Ehrlich in 1872-1874 (Drews, 2000). Ehrlich's work with trypanosomal infections in 1904 led to the introduction of chemotherapy (Bosch and Rosich, 2008). Langley

published the concept of receptive substances in 1905, which introduced a new concept in pharmacology where the receptors function as a "switch" to receive and generate signals that can be either blocked by antagonists or turned on by agonists (Langley, 1905). In 1908, Ehrlich discovered one of the first rational synthetic drugs, arsphenamine, used to cure syphilis. Ehrlich also popularized the notion of the "magic bullet" – the idea that a compound could kill a specific pathogen and not the patient.

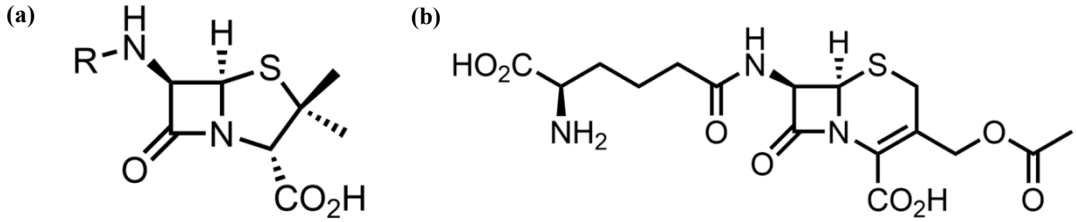


Figure 1.3: Structures of **(a)** penicillin and **(b)** cephalosporin. The characteristic four-membered β-lactam ring can be seen in both structures.

The second period started around the early 20th century when new drugs were discovered, which led to a new era of antibiotic discovery (Pina *et al.*, 2009). This new era is exemplified by the discovery of penicillin (Figure 1.3(a)) by Alexander Fleming in 1929, which opened the door to the treatment of bacterial infections (Fleming, 1929). Chain, Florey, and their collaborators found that some of the antibiotics based on penicillin can be degraded by protective enzymes from bacterial cells (Lobanovska and Pilla, 2017). To prevent bacterial degradation, new antibiotics were discovered by modifying the basic structure of penicillin. Brotzu reported cephalosporin (Figure 1.3(b)) in 1948, which was used to treat infections that were resistant to penicillin by modifying the β-lactam core (Bo, 2000). By then, many pharmaceutical companies such as Merck, Sandoz, and Takeda increased their microbiology facilities to find drugs with other pharmacological or therapeutic properties. Examples include Ivermectin (drug against tropical

filariosis, Figure 1.4(a)) and lovastatin (hydroxylmethylglutaryl-coenzyme A (HMG-CoA) reductase inhibitor, Figure 1.4(b)) (Drews, 2000).

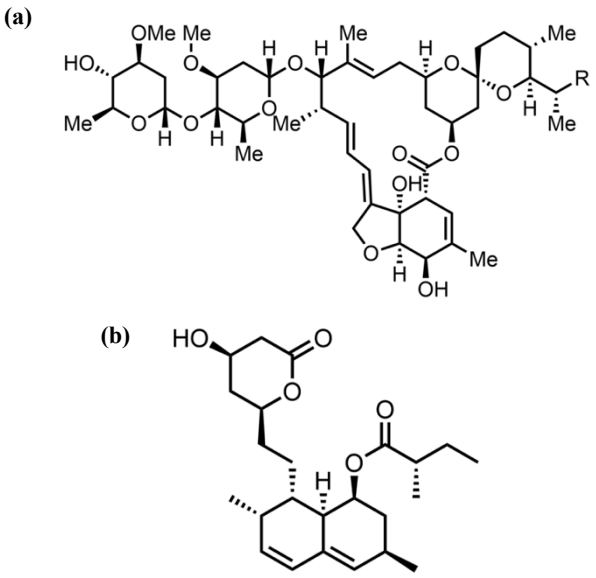


Figure 1.4: Structures of (a) Ivermectin and (b) lovastatin.

In 1953, building on the work of Franklin, Watson and Crick discovered the structure of deoxyribose nucleic acid (DNA), which acted as the foundation for all the complex genetic research techniques and procedures that we use to this day (Watson and Crick, 1953). DNA polymerases are essential enzymes for DNA replication, repair, and recombination. The high number of possible candidates for DNA polymerases created the necessity of Quantitative structure-activity relationship (QSAR) to guide the DNA polymerase inhibitors (García *et al.*, 2011).

QSAR was the first step in the story of the development of AI-based drug discovery. QSAR introduced by Hansch and Fujita in 1964 is a ligand-based drug design method that is an efficient method for building predictive models of molecular properties from structural features (Hansch

and Fujita, 1964). These models attempt to find a statistical correlation between chemical structure and continuous biological or toxicological properties. The earliest QSAR models used regression and classification techniques (Cherkasov *et al.*, 2014). Properties such as half maximal inhibitory concentration (pIC_{50}, Equation 1.1) and half maximal effective concentration (pEC_{50}, Equation 1.2) are logarithmically transformed values of IC_{50} and EC_{50}.

$$pIC_{50} = -\log_{10} IC_{50} \tag{1.1}$$

$$pEC_{50} = -\log_{10} EC_{50} \tag{1.2}$$

IC_{50} indicates the concentration of an inhibitor needed to inhibit a given biological process or component *in vitro* by 50% (Aykul and Martinez-Hackert, 2016); whereas EC_{50} represents the concentration of drug required to obtain 50% of the maximum response *in vivo* (Brooks *et al.*, 2019).

In 1970, the concept of molecular cloning, which applies to the creation of recombinant DNA molecules spurred the progress in the life sciences (Ashwini *et al.*, 2016). This enabled rapid identification of biological targets and their expression with sufficient purity and quantities adequate for structural determination. The emergence of public protein databases such as Protein Data Bank (PDB) and Universal Protein Resource (UniProt) aided in small-molecule drug discovery (Pina *et al.*, 2009). These databases provided a rich source of information for QSAR. Protein's 3D structures can give great insight into how small-molecule ligands bind to their target proteins (Burley, 2021). These structural data have also proven useful in converting biochemically active compounds into potent drug-like molecules (Stoll *et al.*, 2011).

Towards the end of the 20th century, rapid advances in drug discovery occurred due to the development of new techniques such as molecular graphics and modelling, automated high-

throughput screening, and combinatorial chemistry (Pina *et al.*, 2009). Recombinant DNA technology revolutionized the pharmaceutical industry by enabling the identification of potential drug targets. By the completion of the Human Genome Project in 2003, molecular pathways responsible for certain diseases were better understood and acted as a starting point to search new drugs based on proteins.

In the realm of biologics, monoclonal antibodies were introduced as a therapeutic due to the technological breakthrough caused by the hybridoma technology in 1975 (Gershell and Atkins, 2003). Therefore, a transition period occurred between the 20th and 21st centuries when molecular biology became increasingly associated with drug discovery (Pina *et al.*, 2009). Natural proteins were replaced with recombinant versions by genetic engineering. The first product of DNA recombinant technology, which was approved by Food and Drug Administration (FDA) for human therapeutic use, was human insulin in 1982 (Kayser and Warzecha, 2012).

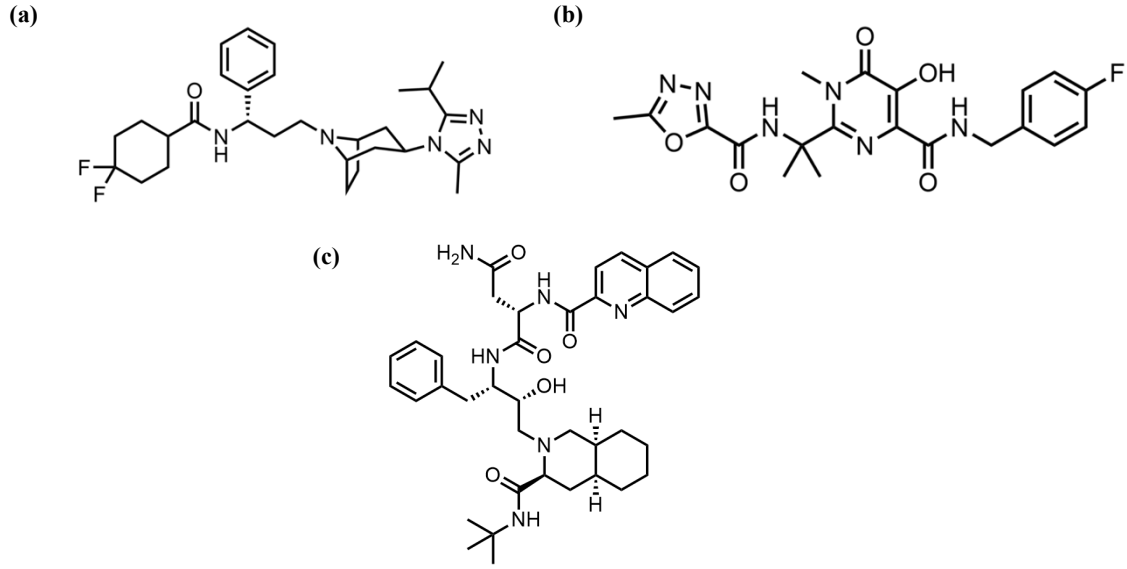


Figure 1.5: Structures of **(a)** maraviroc, **(b)** raltegravir, and **(c)** saquinavir.

By 2002, 120 protein-derived biopharmaceuticals obtained approval in FDA and/or European Medicines Evaluation Agency (EMEA) for therapeutic purposes (*e.g.*, blood factors such as factors VIII and IX) (Walsh, 2003). During this period, new types of oral anti-human immunodeficiency virus (HIV) drugs such as maraviroc (Figure 1.5(a), Pfizer) and raltegravir (Figure 1.5(b), Merck) were approved in 2007 by FDA to treat HIV-1. In addition, new molecular entities of temsirolimus (Figure 1.6, Wyeth) were approved in 2007 by FDA and EMEA for the treatment of advanced renal cell carcinoma (B. Hughes, 2008). The first HIV-1 protease inhibitor approved by the FDA was saquinavir (Figure 1.5(c), Roche) in 1995.

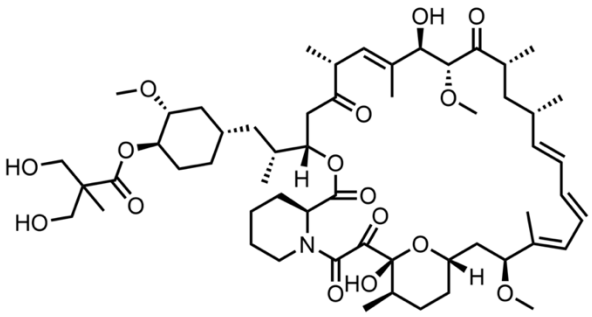


Figure 1.6: Structure of temsirolimus approved by FDA and EMEA in 2007.

Biopharmaceutical industries introduced biogenerics where products are very close in structure and function to an approved biopharmaceutical (Tucker, 2008). In 2006, the first biogeneric received approval from EMEA for omnitrope (Sandoz), which is a biogeneric follow-on from somatropin (Pfizer) approved by FDA (Lawrence, 2007). Embryonic stem cells equally contributed to drug development by their abilities of self-renewal and differentiation into virtually any cell type *in vitro* (Sartipy *et al.*, 2007). The first human embryonic stem cells were isolated from human blastocysts in 1998 (Thomson *et al.*, 1998). In 2005, only 20 new drugs were approved by the FDA, compared to 40 new drugs in 2004. Human stem cell technologies have been used to assist clinical trials. By using this technology, the drug discovery process is made

more efficient with a decrease in *in vivo* testing (Sartipy *et al.*, 2007) and can reveal toxicity of certain drugs at an earlier stage in clinical trials, which avoids conventional animal testing (Keller, 2005).

Experimental techniques such as X-ray diffraction and nuclear magnetic resonance (NMR) are also important for the prediction of the 3D structure of an expressed and purified target protein, either alone or when complexed with a specific drug (Edwards *et al.*, 2000). NMR spectroscopy has also proved to be effective in identifying multivariate metabolites (Jain, 2002). This improves the lead compound selection in drug toxicity screening. The "omics" revolution in the 21st century also contributed towards an increase in biopharmaceutical drugs for therapeutic use, with over 165 biopharmaceutical products (monoclonal antibodies, recombinant proteins, and nucleic acid-based drugs) meeting the FDA and EMEA approvals in 2009 (Pina *et al.*, 2009). As reported by the FDA in 2021, there are currently over 20,000 prescription drug products approved for marketing (FDA, 2021). Despite these successes, pharmaceutical research and development (R&D) is currently facing a decline in productivity in terms of New Molecular Entities (NMEs), as R&D investments are concentrated in areas with high failure rates, which potentially leads to lower post-launch competition and higher sales (Pammolli *et al.*, 2011).

Drug discovery begins with "target identification" where the activation or inhibition of a biological entity (*e.g.*, a protein), part of a key pathway that induces a therapeutic effect on a disease, is identified (Figure 1.7) (J. P. Hughes *et al.*, 2011). Although this thesis focuses on this target-based approach, there are other types of drug discovery such as phenotypic drug discovery where a target does not need to be identified. In this case, the drug candidates are assessed based

on their therapeutic effects on the disease model, for instance, in cells, tissues or organisms (Moffat *et al.*, 2017).

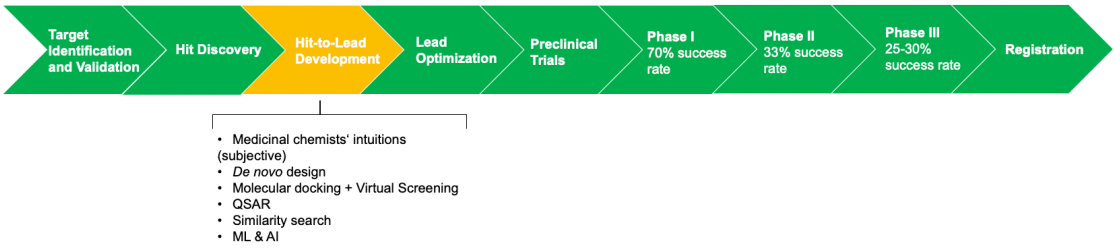


Figure 1.7: The pipeline in drug discovery from target identification and validation to registration. Hit-to-lead development is highlighted in yellow, since this stage is the focus of this thesis. This stage was traditionally driven by medicinal chemists based on their own chemical intuition from experience and ease of synthesis, which are both subject to cognitive bias. There are several computational approaches that exist (*de novo* design, molecular docking, QSAR, similarity search, and applications of machine learning (ML) and AI), which can be used as an alternative, but there are also other approaches.

After the target is identified, further validation of the assay and the target may be required before progressing to "hit identification and lead discovery" phase (J. P. Hughes *et al.*, 2011). This phase requires an intensive search to find a small molecule or biological therapeutic. In small molecule drug discovery, a 'hit' molecule is identified after the target is selected and validated. This 'hit' molecule is a compound with desired activity during compound screening. The number of hits may be reduced via triaging before moving to the next phase by considering factors such as novelty, structural diversity, patentability, and potency.

In the "hit-to-lead development" phase, each hit is chemically refined to improve on properties such as potency, selectivity against a chosen target, physicochemical properties, and efficacy *in vivo*. During the "lead optimization" phase, the desired properties in lead compounds need to be maintained whilst improving lead structure's deficiencies. Factors such as pharmacokinetics,

pharmacodynamics, dose linearity (*i.e.*, relates dose to biological system's response), and metabolic profiling are all conducted by this stage. In "preclinical trials", candidates must satisfy factors such as safety, pharmacokinetics, pharmacodynamics, and process development (*i.e.*, costs for scalability and environmental concerns for manufacturing) through *in vitro* and *in vivo* testing (Zanders, 2020). Toxicity must be studied closely to determine the safe ranges of dosage that are needed. Therefore, a large number of criteria must be considered in this simultaneous optimization process.

Once a reasonable compromise in the properties has been found, the candidate lead(s) can progress into "clinical trials", which consists of three phases. "Phase I" involves finding the maximum dose that can be tolerated by humans without compromising safety (typically, 20 to 100 healthy volunteers) and the FDA reports that approximately 70% of drug candidates move to the next phase (FDA, 2018). "Phase II" involves testing for efficacy on patients with the disease that the drug candidate is designed to treat (up to several hundred people with the disease/condition); the FDA reports that approximately 33% of the drugs that make it into Phase II, move to the next phase. Phase III involves monitoring for any safety issues that may arise over a period of several years and testing for efficacy (300 to 3,000 volunteers who have the disease or condition). The drug candidate is often compared with the current treatment of practice (*i.e.*, existing drug) and larger samples compared to Phase II ensure more statistically significant evidence of the effectiveness of the candidate. It is not uncommon for candidates that looked promising in Phase II to show no efficacy in Phase III (Harrison, 2016; D. Sun *et al.*, 2022). The FDA reports that approximately 25-30% of the drugs that make it into Phase III move to the final stage.

During the "registration" stage, a drug candidate is typically approved as a true 'drug'. The companies may conduct further clinical trials (Phase III) to extend the label for increased specificity and definition (Nwaka and Ridley, 2003). For example, the drug which is approved for disease A is also tested on disease B to check for its efficacy (*e.g.*, cancer drugs). Other factors such as monitoring the safety profiles on different patient groups, dealing with side effects as well as marketing authorization fall under this stage (Zanders, 2020).

Therefore, the goal of molecular drug discovery is to find novel molecules that are not already covered by patents that also have the desired therapeutic properties. Traditionally, medicinal chemists have relied on their own chemical intuition to design small molecule drugs. One of the difficulties is that multiple criteria, such as safety, bioavailability, *etc.*, must be simultaneously satisfied in order to develop a successful drug: an example of a classic multi-objective optimization problem (J. P. Hughes *et al.*, 2011). The kinds of molecules that are proposed by medicinal chemists are often biased by experience, and what is most reliable and quickest to synthesize. There is evidence stating that this approach is unsystematic and narrow in scope (Kutchukian *et al.*, 2012).

Pharmaceutical companies typically devote 12-15 years' worth of work (Nicolotti *et al.*, 2011) and it is currently estimated to spend three billion US dollars to find a new drug (DiMasi *et al.*, 2016). Despite these efforts, Wong *et al.* have estimated that the overall probability of success for clinical trials of biological drugs and vaccines is 13.8% (Wong *et al.*, 2019), but for small molecule drugs it is 5% (KMR-Group, 2012). This difference may be because biologics are much more specific than small molecule drugs. Therefore, there is less chance of side effects or binding to off-targets. In order to improve the latter's odds, new computational methods continue to be

developed. One family of algorithms is used to generate new molecules and are collectively known as "*de novo*" methods, which is the focus of this thesis; they have been extensively studied and proven to be promising (Amaravadhi *et al.*, 2014; Gómez-Bombarelli *et al.*, 2018; Hartenfeller and Schneider, 2011; Kadurin, Aliper*, et al.*, 2017; Nicolaou *et al.*, 2012; G. Schneider and Fechner, 2005; Yasonik, 2020). Finding an efficient method for exploring drug-like chemical space, which is estimated to be on the order of 10^{60}-10^{100} small molecules has been of great interest to researchers and pharmaceutical companies (Drew *et al.*, 2012). This is because synthesis and experimental screening of all chemical space for bioactive molecules is impossible. Consequently, a more systematic and unbiased way of exploring chemical space than a medicinal chemist's intuition is needed.

1.3 *De novo* Drug Design

De novo drug design refers to the usage of computational algorithms to design novel molecules, which can satisfy a range of constraints (P. Schneider and Schneider, 2016). The term "*de novo*" means "from the beginning", which indicates that one can generate novel molecular entities without starting from a template (Devi *et al.*, 2015). The concept of *de novo* drug design was introduced approximately 40 years ago (Danziger and Dean, 1989; Goodford, 1984).

There are three main components of *de novo* drug design: (i) description of the receptor's active site or the ligand's desired pharmacophore, (ii) generation of the molecules, and (iii) evaluation of the generated molecules. *De novo* drug design methods fall into two major categories (Figure 1.8), structure-based and ligand-based designs (Mouchlis *et al.*, 2021).

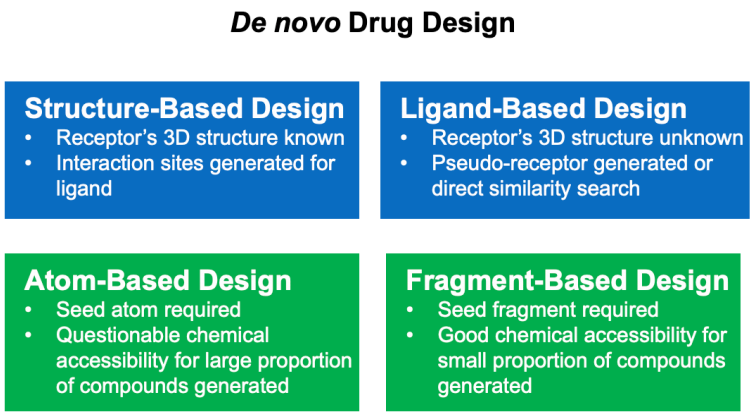


Figure 1.8: Schematic representation of *de novo* drug design methods including major categories (blue) and sub-categories (green).

In structure-based *de novo* drug design, typically the 3D structure of the receptor is known via X-ray crystallography, NMR, or cryo-electron microscopy (Berman *et al.*, 2000), and potential molecules are evaluated by considering the interactions between the ligand and the receptor in the desired binding site. In the case of ligand-based *de novo* drug design, when the 3D-structure of the receptor is unknown, the drug candidate is evaluated using knowledge derived from compounds with known activity.

Atom-based and fragment-based *de novo* drug design methods are sub-categories of the *de novo* drug design methods mentioned earlier (Figure 1.8). In atom-based *de novo* drug design, a single atom can act as a seed to generate new molecules and since the modification of molecules takes place at an atomic-scale, it can result in more exploration of local chemical space (*e.g.,* LEGEND (Nishibata and Itai, 1991)). LEGEND automatically generates drug candidate structures based on a known receptor structure. In this method, compounds that fit well into a receptor cavity are generated by adding atoms one-by-one using a force field and random numbers (Devi *et al.*, 2015; Nishibata and Itai, 1991). In LEGEND, an anchor atom is randomly selected from several

hydrogen-bonding heteroatoms in the protein. The position and atom type of the ligand's first atom are found to create a hydrogen bond to the anchor atom. The subsequent atoms, atom types, and their bond types (up to the user's specified number of atoms) are generated randomly to build new compounds. The values of the bond length and bond angles are assigned based on an array of related atom types from the MM2 force field and parameters (Allinger, 1977). LEGEND ends by adding missing carbon atoms to aromatic rings and supplies hydrogen atoms for all remaining valences of all non-hydrogen atoms. The structure is optimized using van der Waals and electrostatic energies based on atomic charges calculated by the Del Re's method (Del Re, 1958).

Fragment-based *de novo* drug design methods generate new molecules by assembling fragments to propose compounds that offer diverse hits, save experimental cost, and increase ways to develop novel compounds (Erlanson *et al.*, 2016). Examples of fragment-based *de novo* drug design methods include CONCERTS (Pearlman and Murcko, 1996) and LUDI (Böhm, 1992). In CONCERTS (creation of novel compounds by evaluation of residues at target sites), a set of fragments defined by the user can move independently in a target's active site during molecular dynamics simulation. In this way, it enables the fragments to sample low energy orientation. Bonds can be formed between fragments when the geometry between nearby fragments is appropriate, which leads to larger molecules. A key innovation of CONCERTS is the covalent connections that are made amongst the particles in a stochastic and dynamically reversible manner. The changes in topology are either accepted or rejected depending on their ability to improve the total energy of the protein-ligand complex.

LUDI defines the new set of rules for possible non-bonded contacts between a ligand and protein using a library of more than 500 known fragments. In LUDI, a new substituent can be added to

an existing ligand. LUDI works by first calculating the interaction sites on the protein surface to form hydrogen bonds or to fill a hydrophobic pocket. The interaction sites are found from distributions of non-bonded contacts, which are generated from a search in Cambridge Structural Database. In the second step, molecule fragments get mapped to the interaction sites. In the final step, LUDI connects fragments using bridges to form a single molecule. Examples of bridge fragments are -CH_2-, -CH_2CH_2-, -O-, -CO-, -NH-, and -SO_2-. One of the biggest challenges in *de novo* drug design is that the synthetic accessibility of the generated molecules is not guaranteed (G. Schneider and Fechner, 2005).

1.4 Ligand-Based *De Novo* Drug Design

Ligand-based *de novo* drug design is useful when the structure of the target receptor is unavailable (Wise *et al.*, 2002). LeapFrog is one of the early ligand-based *de novo* drug design methods where the algorithm generates new compounds starting from known ligands. This is achieved through the combination of genetic algorithm and 3D molecular field scoring model of comparative molecular field analysis (3D-CoMFA) (Cramer *et al.*, 1988; Payne and Glen, 1993). SPROUT was another pioneering ligand-based method where novel compounds are designed using atomic constraints derived from known drugs/ligands with presence or absence of 3D coordinates of the receptor (Gillet *et al.*, 1995; Gillet *et al.*, 1994). SPROUT was one of the first approaches that addressed synthetic feasibility of designed compounds through Computer Assisted Estimation of Synthetic Accessibility (CAESA). CAESA ranks structures according to a calculated estimate of synthetic ease. Prior to SPROUT, almost all the early *de novo* drug design methods were generating chemically valid molecules by following a simple rule of atom valences and bond orders without considering chemical feasibility.

TOPology-Assigning System (TOPAS) attempts to overcome the limitation of chemically valid molecules by addressing the generation of unrealistic compounds from the combination of small fragments. In TOPAS, a set of rules is derived from retrosynthetic fragmentation of a database containing biologically active ligands (G. Schneider *et al.*, 2000). Novel compounds are then generated by recombining fragments using genetic algorithm with scoring function based on pharmacophore similarity with respect to a relevant compound. Vast exploration of chemical space is promoted by a relatively fast pharmacophore similarity calculation.

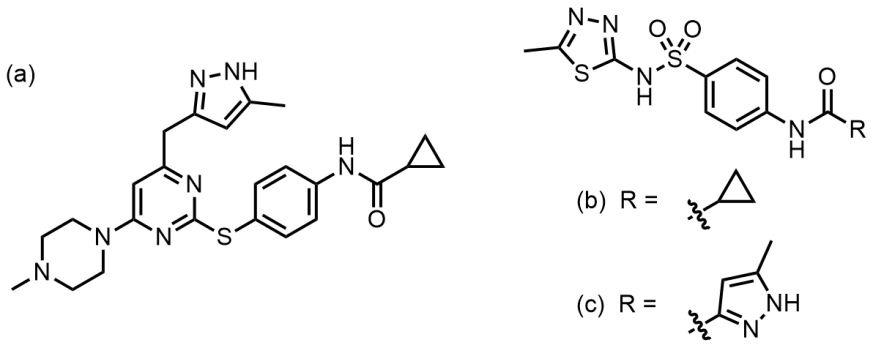


Figure 1.9: Development of AurA inhibitors from VX-680 (a) as a reference for DOGS. *De novo* designed compounds are (b) and (c).

Design of Genuine Structures (DOGS) extended the concepts from TOPAS (Hartenfeller *et al.*, 2012). In this method, molecules were generated using a set of approximately 21,000 available synthetic building blocks and 58 established reaction schemes. This feature allowed DOGS to suggest a proposed synthetic route for each designed molecule, which mimics the work of medicinal chemists. DOGS was used to generate novel Aurora A kinase (AurA) inhibitors, a validated drug target against several types of cancer (Rodrigues *et al.*, 2013). Rodrigues *et al.* used VX-680 (Figure 1.9(a)), which is a *pan*-Aurora inhibitor with high potency against AurA (K_i = 0.6 nM), but its development was abandoned during clinical trials due to QT interval

prolongation (*i.e.*, related to heart rate response) (Harrington *et al.*, 2004). By using Figure 1.9(a) as a template, DOGS was employed, and *de novo* designed compounds of Figures 1.9(b) and 1.9(c) were selected for synthesis.

1.5 Structure-Based *De Novo* Drug Design

A binding site of a protein receptor dictates the possible non-covalent intermolecular interactions between the ligand and receptor atoms and thereby imposes shape constraints on the ligand (G. Schneider and Fechner, 2005). These non-covalent interactions include hydrogen bonds as well as electrostatic and hydrophobic interactions (Mouchlis *et al.*, 2021). The constraints imposed by a binding site can promote selectivity by reducing the number of suitable generated molecules. GRID (Goodford, 1985) and HSITE (Danziger and Dean, 1989) were two of the first developed *de novo* design algorithms, which are able to map protein cavities in order to identify potential ligand interaction points. These interaction points were then used as a guide for *de novo* design. These two methods could also automatically generate an array of data points, geometrically consistent in angle and distances that are typical for small molecules, and suitable for further elaborations of the generated ligand core structures. At protein-ligand interaction points, both methods can identify potential hydrogen bonds. GRID was also capable of mapping other molecular features including aromatic and lipophilic features.

PRO-LIGAND (D. E. Clark *et al.*, 1995) is another structure-based *de novo* drug design method that consider hydrophobic interaction regions. This method incorporates many novel features for fragment placing, fragment assembly, and prescreening of library fragments. GrowMol is a grid-based *de novo* design method, where the points in the receptor's active site are defined using grids and the hydrophobic or hydrogen bonding interactions are calculated at each grid point (Bohacek

et al., 1999). This method generates organic molecules that are both spatially and chemically complementary to the target protein's binding site. At each step, the position and atom type that are added get randomly selected using Boltzmann statistics to create a favorable bias for the atoms that can form favorable interactions with the binding site residues.

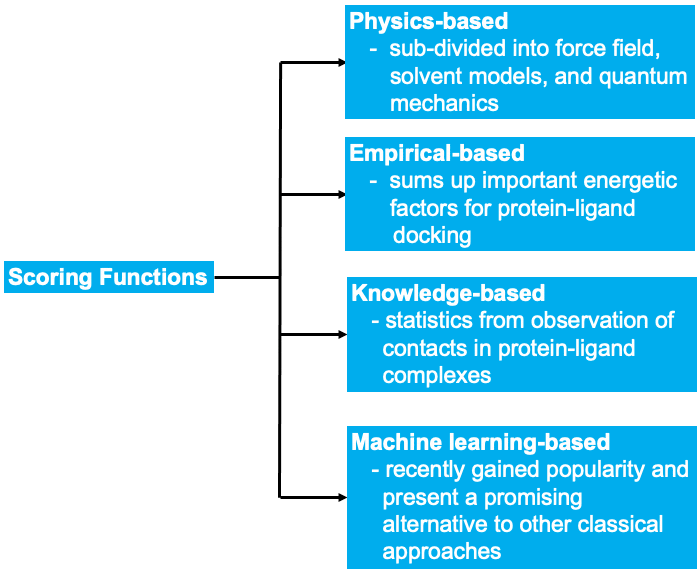


Figure 1.10: Categories of scoring functions including physics-based (force field, solvent models, and quantum mechanics), empirical-based, knowledge-based, or machine learning-based.

In structure-based *de novo* drug design, calculations of the predicted free energy of binding of the proposed small molecule and receptor are more accurate, but rapid scoring functions can be used to evaluate large libraries of compounds (Liu *et al.*, 1999; Luo *et al.*, 1996; Zhu *et al.*, 2001). These scoring functions can be categorized as physics-based (force field, solvent models, and quantum mechanics), empirical-based, knowledge-based, or machine learning-based (Figure 1.10).

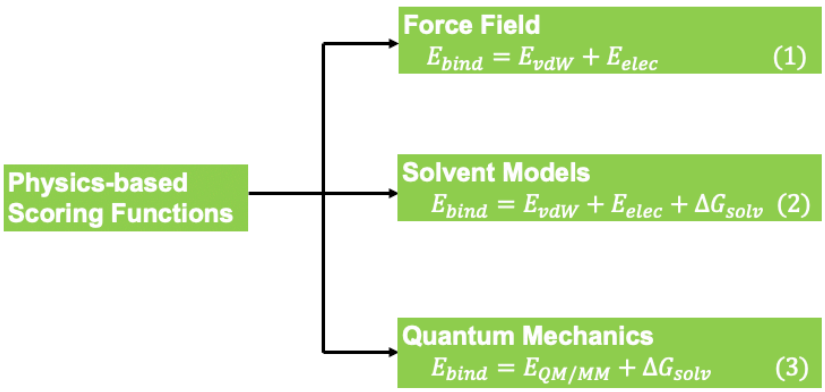


Figure 1.11: Classification of physics-based scoring functions including force field-based, solvent models, and quantum mechanical-based. E_{bind} stand for binding energy. Force field-based scoring function is a summation of van der Waals (E_{vdW}) and electrostatic interaction (E_{elec}). Binding energy in solvent models are found from E_{bind} in force field-based scoring function plus solvation/desolvation effect (ΔG_{solv}). Quantum mechanics-based scoring function is a summation of quantum mechanical/molecular mechanics (QM/MM) energies and ΔG_{solv}.

Physics-based scoring functions are classified into force field-based, solvation models, and quantum mechanical-based (Meng *et al.*, 1992). The force field-based scoring function computes the binding energy by considering the van der Waals (E_{vdW}) and electrostatic interaction (E_{elec}) between the protein-ligand atom pairs (Equation 1 in Figure 1.11), which considers the contribution of enthalpy to energy. Since the force field-based scoring function neglects entropy and solvent effect, the performance of this type of scoring function is not sufficient (Huang *et al.*, 2010). The force field-based scoring function is improved by incorporating the torsion entropy of ligands and the solvation/desolvation effect (ΔG_{solv}) described by explicit (Michel *et al.*, 2009; Y. Yang *et al.*, 2013) and implicit solvent models (H. Sun *et al.*, 2013; H. Y. Sun *et al.*, 2014) (Equation 2 in Figure 1.11). However, the predictive accuracy of the binding energy is significantly dependent on the functional form of the potential energy and related parameters that are hard to locate. This is because the scoring function for solvent models is based on force field-based scoring function. Quantum mechanical-based scoring functions were developed to overcome the challenges of covalent interactions, polarization, and charge transfer in docking

(Kulik, 2018; Orozco-Gonzalez *et al.*, 2017). Although this type of scoring function has more accuracy than the force field-based scoring function, the computational expense is high. Therefore, a hybrid quantum mechanical/molecular mechanics (QM/MM) approach was developed to compromise predictive accuracy and computational cost (Equation 3 in Figure 1.11) (Raha *et al.*, 2007).

Empirical-based scoring functions (Eldridge *et al.*, 1997; Murray *et al.*, 1998) are based on the binding affinity of a complex by summing up the important energetic factors for protein-ligand docking (*e.g.*, hydrogen bonds, hydrophobic effects, and steric clashes), where the coefficients are determined by regression analysis using experimentally derived binding affinities (Kadukova and Grudinin, 2017). Knowledge-based scoring functions (Gohlke *et al.*, 2000) are based on statistics made from observations of contacts from crystal structures of protein-ligand complexes. This scoring function will score a frequently seen contact more favorably. The greatest advantage of knowledge-based scoring functions over physics-based and empirical-based scoring functions is that it compromises for computational cost and predictive accuracy (Huang and Zou, 2010). However, the reference state for the knowledge-based scoring functions is difficult to locate.

Machine learning-based scoring functions have recently gained popularity and present a promising alternative to other three classical scoring functions discussed above (Shen *et al.*, 2019). Machine learning-based scoring functions are often reported to outperform classical approaches since these methods may offer an advantage by identifying non-linear relationships between the binding affinity and protein-ligand complex features (Khamis *et al.*, 2015). However, disadvantages include the over-fitting of the model and the lack of interpretability of the black box model (Shen *et al.*, 2019).

1.6 Machine Learning and AI

The field of AI grew out of attempts to simulate human intelligence, to mimic behaviors such as natural language processing, visual perception, knowledge representation, learning, pattern recognition, problem solving, intelligent robots, *etc.* by using computers (Figure 1.12 (grey)) (H. C. S. Chan *et al.*, 2019; S. Li *et al.*, 2021; X. Yang *et al.*, 2019).

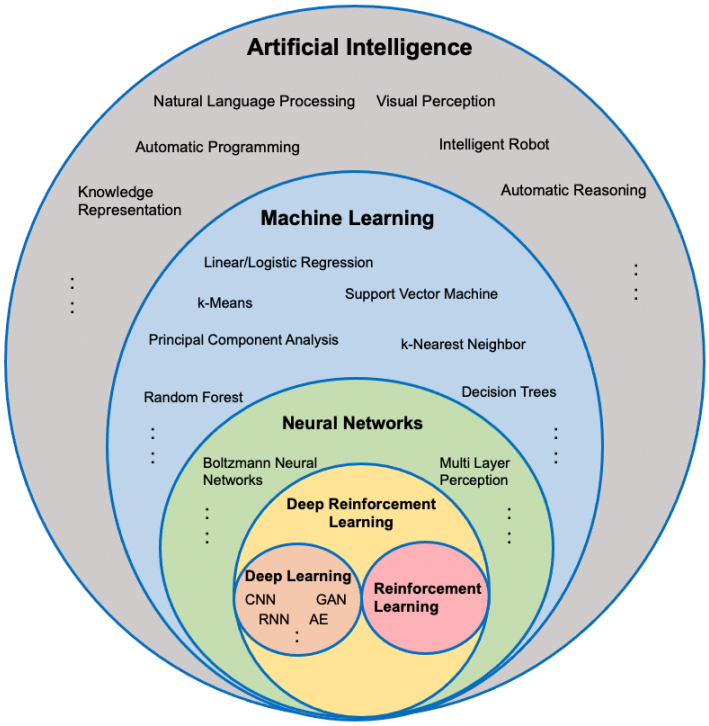


Figure 1.12: Schematic of relationships between Artificial Intelligence (grey), Machine Learning (blue), Neural Networks (green), Deep Reinforcement Learning (yellow), Deep Learning (orange), and Reinforcement Learning (pink). CNN, GAN, RNN, and AE stand for convolutional neural networks, generative adversarial networks, recurrent neural networks, and autoencoders, respectively.

In supervised learning, a model is built from labelled training data with independent variables, $X = x_1, x_2, \ldots, x_n$, and output or dependent variables, $Y = y, y_2, \ldots, y_n$ (Russell and Norvig, 2009). The model attempts to learn the mapping function, f (Equation 1.3).

$$Y = f(X) \tag{1.3}$$

The activity data used in supervised learning needs to be split into a training, validation, and testing set. Generally, the model is built up during training by iteratively adjusting its parameters to minimize the difference between the predicted and actual values. During validation, the model is evaluated using the validation set and the hyperparameters of the model are fine-tuned. This validation step is also used to indicate whether a model is overfitting or not. In testing, the trained model is used to predict values for the test set and the trained model is evaluated using previously unseen data.

Supervised machine learning can be divided into classification and regression methods as briefly mentioned in Section 1.2. Regression maps a set of input variables to a continuous output variable, such as pIC_{50} in Equation 1.1. Classification maps a set of input variables to discrete or categorical values, such as active or inactive. Examples of supervised machine learning methods include random forests (Breiman, 2001), support vector machines (Cortes and Vapnik, 1995), and artificial neural networks (Fausett, 1994).

In unsupervised learning, a model is built from unlabeled data (Murphy, 2012). Therefore, one of the widely used unsupervised learning methods of clustering is employed to find underlying structure in the data. Clustering creates groups based on similarities in molecular descriptor (Leach and Gillet, 2007). This implies that compounds within the same cluster will share similar molecular descriptors, whereas compounds in different clusters will share dissimilar molecular descriptors. The general first step in clustering is to generate molecular descriptors for each compound. The second step is to calculate the similarity between all compounds to create a similarity matrix. The third step is to use a clustering algorithm to generate clusters and the final step is to choose a subset of molecules to represent each cluster. By following these steps, the

resulting subset will be smaller than the original dataset but with guaranteed chemical diversity. An example of clustering algorithm used in unsupervised learning is *k*-means clustering. In this algorithm, the user specifies the number of clusters, *k*, to create. Then *k* seed molecules are randomly selected, and the rest of the molecules are assigned to the seed that is closest, which leads to the formation of clusters. The centroid of each cluster is calculated, and molecules are re-assigned to the closest centroid. This centroid calculation process and molecule relocation is repeated until there is no change in centroid molecules.

Semi-supervised learning is a branch of machine learning that aims to combine supervised and unsupervised learning (van Engelen and Hoos, 2020). Typically, semi-supervised learning algorithms improve performance in either classification or clustering problems by using information associated with the other. For a classification problem, additional data points with an unknown label may be used to aid the process. For a clustering method, the learning procedure may benefit from the knowledge that certain data points belong to the same class.

Reinforcement learning (Figure 1.12 (pink)) is another branch of machine learning that explores a series of actions to maximize a desired objective called the reward (Kaelbling *et al.*, 1996). The benefit of using reinforcement learning over other branches in machine learning is that it does not require a dataset to be trained on, and since it operates in a dynamic environment, it can "learn" adaptively from experience. As the agent performs each action in its environment, it collects new data. The actions taken by the reinforcement learning model are defined by a policy and each action results in a penalty or reward (Section 1.7). In general, favorable actions are given greater reward or less penalty than unfavorable ones. The learning takes place by finding the optimal

policy, which selects the action at each state whilst minimizing the temporal difference (TD) error (Section 2.2.4).

Neural networks (Figure 1.12 (green)) derived deep reinforcement learning (Figure 1.12 (yellow)), deep learning (Figure 1.12 (orange)), and reinforcement learning (Figure 1.12 (pink)). This network simulates the mode of human brain processing information and uses gradient descent method and back-propagation algorithm to correct its own parameters automatically to fit the data better (Kriegeskorte and Golan, 2019; LeCun *et al.*, 2015; Zipser and Andersen, 1988). As compared to traditional neural networks, deep learning uses multi-layer neural networks that are more powerful in fitting capabilities (Manisha *et al.*, 2020). Therefore, deep learning (Figure 1.12 (orange)) combines big data with computational power to make multi-layer neural networks possible such as recurrent neural networks (RNN), generative adversarial networks (GAN), convolutional neural networks (CNN), and autoencoders (AE) (Chen *et al.*, 2018). As shown in Figure 1.12 (yellow), deep reinforcement learning combines deep learning (*i.e.*, multi-layer neural networks) together with reinforcement learning, and is now being used in *de novo* drug design (Lavecchia, 2019).

1.7 *De Novo* Drug Design using AI

Improvements in the predictive power of machine learning and deep learning methods have led to a significant growth in interest in these techniques to facilitate the generation of new molecules (Dalke *et al.*, 2018; Zhou *et al.*, 2019). There are four classes of generative models: RNN, AE, GAN, and reinforcement learning (RL) (Wang *et al.*, 2022). In *de novo* drug design, molecular generation are popularly performed using simplified molecular input line entry system (SMILES) representation of chemical molecules, which is a string-based representation derived from

molecular graphs (Weininger, 1988). Recently, deep generative models have been trained directly on graph representation of molecules, which became a feasible alternative (Xu *et al.*, 2019).

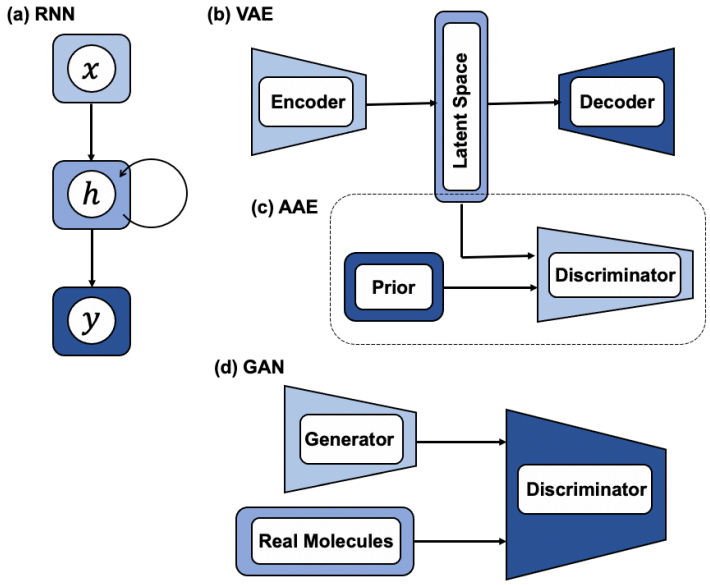


Figure 1.13: Schematic representations of the neural network architectures for *de novo* drug design: **(a)** Recurrent Neural Networks (RNN); **(b)** Variational Autoencoder (VAE); **(c)** Adversarial Autoencoder (AAE); **(d)** Generative Adversarial Network (GAN). VAE is a basic autoencoder, whilst AAE is a variant of VAE.

RNN is one of the most commonly used classes of generative models to design novel compounds. RNN can process string of characters in a sequence of input vectors, $x_{1:n} = [x_1, \ldots, x_n]$ (x in Figure 1.13(a)) (Xu *et al.*, 2019). From an initial hidden state vector h_0 (h in Figure 1.13(a) and h_i in Equation 1.4),

$$h_i = f_r(h_{i-1}, x_i) \tag{1.4}$$

$$y_i = f_0(h_i) \tag{1.5}$$

where h_i is a hidden state vector at time i, which keeps memory of the whole sequence information. f_r and f_0 are recursive and output functions, respectively. Once the model is trained on a dataset of molecules such as ChEMBL (Mendez *et al.*, 2019), ZINC (Irwin *et al.*, 2012), and PubChem (Kim *et al.*, 2021), RNN can be used to generate new characters (y in Figure 1.13(a)

and y_i in Equation 1.5), which leads to novel molecules, based on the probability distribution learned during model training. RNN can be combined with molecular string representations (*e.g.*, SMILES and self-referencing embedded strings (SELFIES)) (Olivecrona *et al.*, 2017; Popova *et al.*, 2018; X. Yang *et al.*, 2017). For example, REINVENT is an RNN-based generative model that performs molecular generation through augmented episodic likelihood-based RL (Olivecrona *et al.*, 2017).

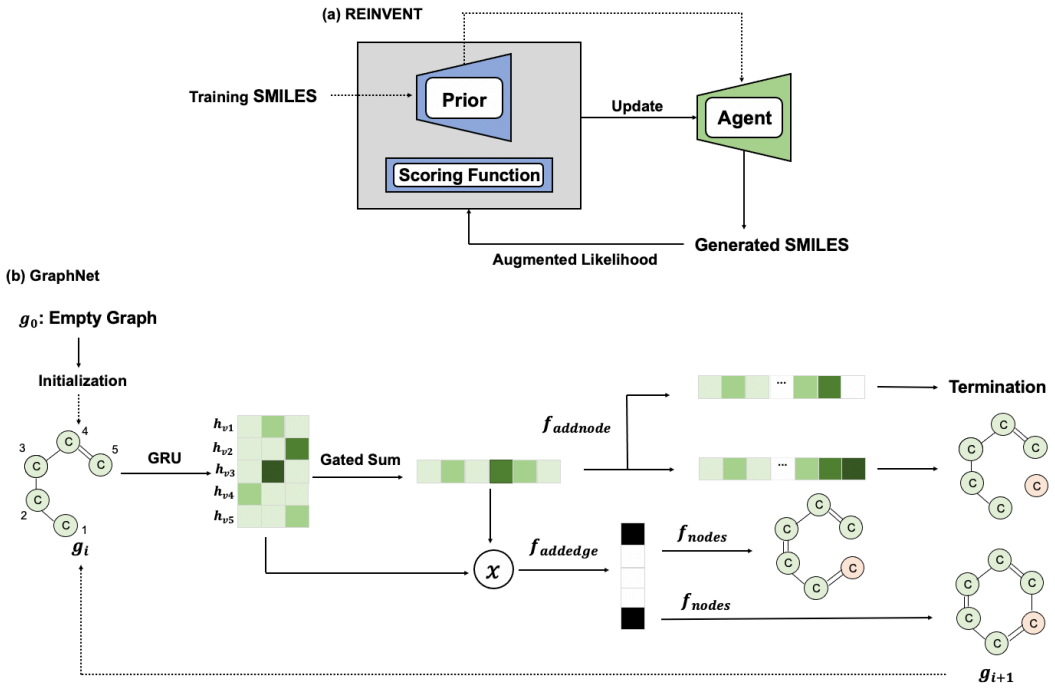


Figure 1.14: Architectures of (a) REINVENT and (b) GraphNet.

In REINVENT, a policy-based RL is used to fine-tune an agent (RNN-based) to generate molecules with desirable properties. "Prior" and "Agent" networks shown in Figure 1.14(a) contain the same RNN-based architecture. The Prior network is composed of three gated recurrent unit (GRU) layers (*i.e.*, a way to store information in flat vectors) that are trained using maximum likelihood estimation (MLE) on the training dataset, so learns both the SMILES syntax and

distribution of molecular structure. The idea behind MLE is that every step in the model leads to a probability distribution over what the next character is likely to be, and the aim is to maximize the likelihood that is assigned to the correct token. Agent network is initialized using the learned probability distributions. The term "augmented likelihood" ($logP(A)_{\mathbb{U}}$ in Equation 1.6) above is defined as a combination of the Prior network ($logP(A)_{Prior}$ in Equation 1.6) and a user-defined scoring function ($S(A)$ in Equation 1.6) for rating the generated sequence (A in Equation 1.6):

$$logP(A)_{\mathbb{U}} = logP(A)_{Prior} + \sigma S(A) \tag{1.6}$$

where σ is a scalar coefficient of the scoring function, S. By learning this augmented likelihood, the Agent network is forced to update its policy for increasing the expected score of the generated molecules whilst preserving the ability of the Prior network. Q_{valid} (Equation 1.7) and Q_{novel} (Equation 1.8) are chemically valid proportion of generated molecules and fraction of novel molecules from generated set, respectively (Xu *et al.*, 2019).

$$Q_{valid} = \frac{|V|}{n_s} \tag{1.7}$$

$$Q_{novel} = \frac{|set(V) \cap X|}{|set(V)|} \tag{1.8}$$

where V is the list of chemically valid molecules, n_s is the number of randomly generated samples, and X is the list of molecules from a specific training set. After training the Prior network on 1.5 million structures from ChEMBL, Q_{valid} (Equation 1.7) and Q_{novel} (Equation 1.8) for REINVENT were 94% and 90% respectively, for 12,800 generated samples of SMILES sequences.

Molecular graphs have also been combined with RNN to design drug-like molecules (Y. Li, Vinyals*, et al.*, 2018; Y. Li, Zhang*, et al.*, 2018). GraphNet is a graph-based RNN where molecular graph is used to create representations of atoms and bonds in a GRU architecture (Y. Li, Vinyals*,*

et al., 2018). GRU performs the information propagation process as shown in Figure 1.14(b). These representations are then used to make sequential graph building decisions. Each of the decision steps is modeled using three modules. First is the $f_{addnode}$ module in Figure 1.14(b) that defines a probability of adding a new node or not. The second is the $f_{addedge}$ module in Figure 1.14(b) that defines the probability of adding a new edge or not. Lastly, f_{nodes} module in Figure 1.14(b) defines a probability of picking one node to connect to the new node with typed edges. These probabilistic decision-making models get parametrized by training with known molecular graphs. During training, fixed ordering and uniform random ordering of node permutation were taken into consideration. In GraphNet, the ChEMBL database was extracted with the limit of at most 20 heavy atoms and the numbers of training, validation, and test set were 130,830, 26,166, and 104,664, respectively. These sets were used to train and test models for GraphNet. For 100,000 generated samples of molecules, Q_{valid} was 97.52% and 95.98%, and Q_{novel} was 90.01% and 95.54% when trained with fixed ordering and random ordering, respectively.

AE is becoming popular in *de novo* drug design, which is used to perform dimensionality reduction (Gómez-Bombarelli *et al.*, 2018; D.P. Kingma and Welling, 2014; D.P. Kingma and Welling, 2019). The basic form of autoencoder is known as variational autoencoder (VAE) as shown in Figure 1.13(b). VAE is designed to learn a probabilistic generative model known as decoder and its posterior known as encoder (Xu *et al.*, 2019). The role of encoder is to transform the input structure into a lower-dimensionality space representation. The decoder is mathematically represented as the probability for generative process of x (the observation) given the continuous latent variable, z, with a likelihood of $p_\theta(x|z)$ where θ is the learnable parameter. The role of decoder on the other hand, is to reconstruct the original input, which essentially performs an inverse operation of encoder. Mathematically, the encoder approximates the posterior

with a model of $q_\phi(z|x)$, which is parametrized by ϕ. The molecular generation in AE takes place in the lower-dimensionality space between these two models (*i.e.*, latent space in Figure 1.13(b)). This is represented as the stage where encoder and decoder are learned simultaneously by maximizing the evidence lower bound (ELBO) of the marginal likelihood as shown in Equation 1.9:

$$ELBO(\phi, \theta) = \mathbb{E}_{q_\phi(z|x)}[log p_\theta(x|z)] - KL\left(q_\phi(z|x) \parallel p(z)\right) \quad (1.9)$$

where θ and ϕ are differential parameters, and KL is the Kullback-Leibler (KL) divergence. The KL term can be interpreted as a regularizer that prevents the inference network from copying x into z. ELBO can be maximized by gradient ascent. Once the model is trained, every new data point which is sampled within the latent space can be translated back into a particular compound by the decoder. In VAE, molecules can be represented as sequences such as in the form of SMILES (Gómez-Bombarelli *et al.*, 2018), SELFIES (Boitreaud *et al.*, 2020), or molecular graphs (Samanta *et al.*, 2020).

Chemical VAE (ChemVAE) is the first VAE-based *de novo* molecular design method introduced in 2018 (Gómez-Bombarelli *et al.*, 2018). As shown in Figure 1.15, ChemVAE consists of an encoder, a decoder, and a predictor. Once the encoder converts SMILES strings into real-valued fix-dimensional continuous vectors (latent space in Figure 1.15), the decoder manages to transform the vectors into SMILES strings. To ensure validity of decoded SMILES strings, ChemVAE adds Gaussian noise to the encoder with penalty term, which guarantees valid decoding. This forces the decoder to learn how to decode points in latent space. A predictive model based on multi-layer perceptron was joined into VAE to predict the molecular properties from latent space using Gaussian Process (GP) or Bayesian Optimization (BO). GP can be used to model any smooth continuous function with few parameters (Rasmussen, 2004). When using

the latent space representation of molecules as an input, the GP gets trained to formulate an acquisition function, which enables evaluation of the target properties and determines what the next point should be. In the case of BO, it is a sequential design strategy for global optimization of black-box functions (Gómez-Bombarelli *et al.*, 2018).

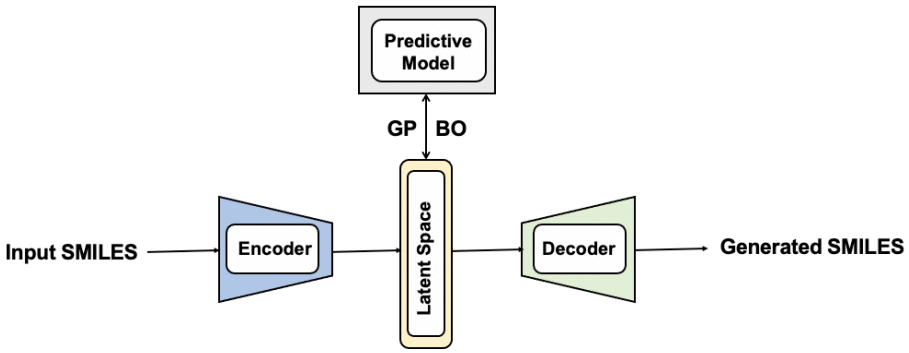


Figure 1.15: Architecture of ChemVAE.

Adversarial Autoencoder (AAE) shown in Figure 1.13(c) is a VAE model (*i.e.*, standard VAE) with the addition of a discriminator network. AAE has also been employed in *de novo* drug design (Kadurin, Nikolenko*, et al.*, 2017; Polykovskiy *et al.*, 2018; Shayakhmetov *et al.*, 2020). AAE is regularized by an adversarial learning (AL) rather than a KL divergence mentioned above for VAE (Xu *et al.*, 2019). In the case of VAE, the KL regularization is imposed to a prior distribution on the latent variable, z. However, for AAE, the AL regularization is used to match the posterior distribution to a prior distribution. The posterior distribution in VAE is usually a Gaussian distribution with mean and variance, which is predicted by the encoder. In the case of AAE, the posterior distribution of $q_\phi(z)$ ("Latent Space" in Figure 1.13(c)) is encouraged to match an arbitrary distribution of prior, $p_z(z)$ ("Prior" in Figure 1.13(c)). Encoder of AE acts as a generator, G, where G tries to fool the discriminator, D, by mimicking $p_z(z)$. In AAE, there are two players

(*i.e.*, generator and discriminator) involved in a minmax game, which is optimized alternatively using stochastic gradient descent as shown in Equation 1.10. The first two terms represent reconstruction, and the last term denotes AL losses.

$$\min_{G,\theta}\max_{D}\mathbb{E}_{x\sim p_{data}(x)}\left[logD\left(G(x)\right)\right]+\mathbb{E}_{x\sim p_z(z)}\left[\log\left(1-D(z)\right)\right]-\mathbb{E}_{x\sim p_{data}(x)}\left[logp_\theta\left(x\middle|G(x)\right)\right] \quad (1.10)$$

where θ is a differential parameter. The objective of discriminator is to maximize the first two terms.

Therefore, during the model training in AAE, this extra network of discriminator forces the low-dimensional latent space to be indistinguishable from a specific prior target distribution. The existence of a discriminator adds improved flexibility to VAE in a drug discovery context (Bian and Xie, 2021). For example, Blaschke *et al*. proposed a SMILES-based AAE molecular generative model, which was built using 1.3 million SMILES strings from the ChEMBL database (Blaschke *et al.*, 2018). In this work, four AE architectures (NoTeacher VAE, Teacher VAE, Gaussian AAE, and Uniform AAE) were used in molecular generation. Both Uniform AAE (78.3%) and Gaussian AAE (77.4%) performed better in Q_{valid} than NoTeacher VAE (19.3%) and Teacher VAE (77.6%). This method was successfully able to reconstruct celecoxib and close analogs demonstrated the smoothest latent space for Uniform AAE as compared to other three AE architectures.

GAN is another neural network architecture used in drug discovery (De Cao and Kipf, 2018; Guimaraes *et al.*, 2017; Sanchez-Lengeling *et al.*, 2017), which also consists of a discriminator as shown in Figure 1.13(d). GAN consists of two main components including a generative model, G and a discriminative model, D (Xu *et al.*, 2019). A generative model learns a map from a prior to the data distribution to sample new data points. A discriminative model learns to classify

whether samples come from the real data distribution rather than from G. These two models are implemented as deep neural networks and trained alternatively with stochastic gradient descent. Since G and D have different objectives, they can be considered two players in a minmax game:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \quad (1.11)$$

where G tries to generate samples to fool the discriminator and D tries to differentiate samples correctly. Therefore, in GAN, the discriminator function distinguishes molecules generated by the model and the ones coming from the training data. This information gets combined with the generative loss function, which drives the model to generate compounds that are indistinguishable from the original training data. Because of its architecture, GAN presents limitations such as unbalance and instable training as well as restriction in learning of chemical space (Bian and Xie, 2021; Elton *et al.*, 2019). However, several approaches in GAN have already been proposed to mitigate some of these limitations (Arjovsky *et al.*, 2017; Kurach *et al.*, 2018). Molecular representations such as SMILES (Guimaraes *et al.*, 2017), SELFIES (Sanchez-Lengeling *et al.*, 2017), and graph (De Cao and Kipf, 2018) can be directly incorporated with GAN. Recently, GAN has been combined with information such as gene expressions and morphological profile images (Méndez-Lucio *et al.*, 2020; Shayakhmetov *et al.*, 2020). This is a new fascinating development in *de novo* drug design since it can lead to generation of new target compounds without any specific knowledge of the biological targets.

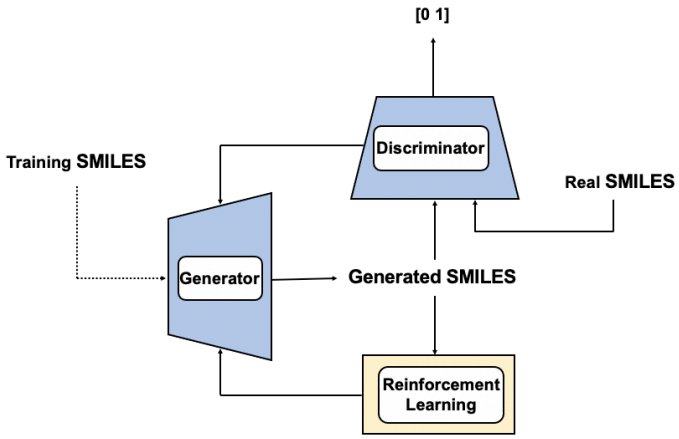


Figure 1.16: Architecture of ORGAN.

Objective reinforced GAN (ORGAN) is a GAN-based generative model combined with reward functions with RL to generate SMILES strings (Guimaraes *et al.*, 2017). As shown in Figure 1.16, "Generator" is trained to fool the "Discriminator" and "Discriminator" classifies real and generated SMILES sequences. "Generator" is trained as an agent through the REINFORCE algorithm and the reward function was supplied by the "Discriminator" and the objectives. Objectives of normalized logarithmic partition coefficient (logP), synthetic accessibility score, and Quantitative Estimate of Druglikeness (QED) were optimized for generating new compounds. ORGAN was trained using 5,000 molecules from the QM9 dataset and showed high Q_{valid} of 88.2-96.5%.

Even though RNNs, AEs, and GANs presented above display powerful approaches for *de novo* drug design, in many cases, these methods design molecules that are not perfectly suited for the drug discovery problem under consideration (Palazzesi and Pozzan, 2022). For example, the selection of model evaluation metrics causes problems for SMILES-based generative models since metrics such as novelty, uniqueness, and diversity of generated molecules are not easy to

compare between different models when applied to different datasets and focused on different reward functions (Xu *et al.*, 2019). Another common problem in SMILES-based generative model is fragment-based growing. Since the fragments of molecules are not well defined in SMILES notation, it is much more difficult to perform fragment-based generation for SMILES-based generative models than the graph-based generative models. In addition, multi-objective reward functions are hard to be trained smoothly in current SMILES-based models since the factors in multi-objective reward functions may be opposed to each other. Graph-based generative models on the other hand usually experience high validity and novelty than the SMILES-based models, but such models often overlook the uniqueness of generated molecules (*e.g.*, GraphVAE (Gómez-Bombarelli *et al.*, 2018) and MolGAN (De Cao and Kipf, 2018)). This is because the reconstruction ability for both generative models often seem to be neglected.

Therefore, other methods have been published to guide the optimization to enable exploration in a desired chemical space (Korshunova *et al.*, 2022). In this attempt, it is worth mentioning RL, which is another class of generative models (Olivecrona *et al.*, 2017; Popova *et al.*, 2018). RL defines a discrete action space, a (Elton *et al.*, 2019). At each time-step, t, the current state of the environment is the sequence generated so far in the form of $(s_0, \dots, s_t)$ and the action, a, is the next token to be chosen defined as $a = s_{t+1}$. The goal of RL is to maximize the expected return of G_T for all possible start states of s_0. The return function of $G_T = \sum_{i=t}^{T} R_i$ sums the rewards over the length of time the agent is active, which is known as an "episode". Mathematically, the optimization problem in RL is expressed as:

$$\max_{\theta} \mathrm{J}(\theta) = \mathbb{E}\left[G_T \mid s_0, \theta\right] \tag{1.12}$$

where θ are the model parameters. In this case, one episode corresponds to the generation of one molecule and there is only one start state. $R_T = 0$ until the end-of-sequence token is generated or

the maximum string length is reached. If T represents the maximum length of SMILES string, then only R_T is non-zero and thus, $G_T = R_T$ for all T. The state transition is deterministic, which means that $p^a_{s,s'} = 1$ for the next state of $s_{1:T+1}$ if the current state is $s_{1:T}$ and the action $a = s_{t+1}$. For other states, s'', $p^a_{s,s'} = 0$. $p^a_{s,s'}$ is the probability of transition from state s to state s' by taking an action a. Due to these simplifications, Equation 1.12 assumes a simple form as shown in Equation 1.13:

$$J(\theta) = R_T \sum_{t=0}^{T} \pi_\theta (a_t|s_t) \tag{1.13}$$

where $\pi_\theta(a_t|s_t)$ is the policy model, which defines the probability for choosing the next action given the current state.

RL is broadly categorized into value learning and policy learning methods. REINFORCE algorithm mentioned earlier is a policy learning method, which is commonly used in RL. This algorithm falls into the class of policy gradient methods where the gradient of $J(\theta)$ in Equation 1.12 is found for a run of length T as shown in Equation 1.14:

$$\nabla J(\theta) = \mathbb{E}\left[G_T \frac{\nabla_\theta \pi_\theta(a_t|y_{1:t-1})}{\pi_\theta(a_t|y_{1:t-1})}\right] \tag{1.14}$$

where $\pi_\theta(a_t|y_{1:t-1})$ is the probability of choosing the next action given the current state. There is a debate regarding superiority of policy learning methods versus value learning methods. Olivercrona *et al.* argues that policy learning methods is a more natural choice for molecular optimization because these methods can start from a pre-trained generative model, whereas value-function learning-based methods cannot (Olivecrona *et al.*, 2017). In addition, most policy learning methods have proven to lead to an optimal policy and the resulting generative models are faster to sample. On the contrary, Zhou *et al.* argues that value function learning methods are superior partly because policy gradient methods suffer from issues with high variance in the

gradient estimates during training (Zhou *et al.*, 2019). Another issue of RL after performing MLE is that the generative model can be drifted too far, which causes an important chemical structural information to be lost leading to highly unstable structures or invalid SMILES (Elton *et al.*, 2019). In Molecule Deep Q-Networks (MolDQN), Zhou *et al.* proposed an approach for molecular optimization to combat this issue, which uses pure RL training instead of using any pre-trained generative model (*e.g.*, RNN) (Zhou *et al.*, 2019). The RL agent directly constructs molecular graphs, which takes an action such as atom/bond addition as well as atom removal.

1.8 Evaluation of *De Novo* Drug Design

There are three main ways of evaluating *de novo* drug design methods (Wang *et al.*, 2022). The first considers the entire generated molecular set by metrics such as validity, novelty, uniqueness, chemical diversity, and chemical space coverage. Chemical space coverage is an important consideration for computational methods in drug discovery, to ensure the most appropriate compounds can be identified before expensive synthesis and testing, so methods to efficiently explore chemical space exploration such as Coverage Score (Woodward *et al.*, 2022) have been developed. The second category is the evaluation of generated molecules by considering metrics such as physicochemical properties of individual molecules (*i.e.*, molecular weight, logP, or topological polar surface area), synthetic accessibility, and QED (Bickerton *et al.*, 2012). The last category consists of benchmarks for evaluating generative models such as GuacaMol (N. Brown *et al.*, 2019) and MOSES (Polykovskiy *et al.*, 2020). There are 20 optimization benchmarks included in GuacaMol for goal-directed models (*i.e.*, models that optimize for a particular reward function) and generative methods are evaluated by the model scores of the generated molecules.

1.9 Future Directions in *De Novo* Drug Design

One of the challenges in *de novo* drug design is that it is expensive and time-consuming to evaluate the quality of the generated molecules using wet-laboratory experiments. The three metric categories described in Section 1.8 are applied instead, but these are not perfect metrics to evaluate *de novo* drug design methods such as the quality of the generated molecules. The standards that quantify the quality of each *de novo* drug design method would aid the advancement of the field. In addition, the selection of the training set is critical for supervised *de novo* drug design methods that require pre-training, but good data is scarce and there is still no standard for evaluating the structural diversity of the training set (although Coverage Score (Woodward *et al.*, 2022) is a step in the right direction). The size of the training set could also contribute to its quality by preventing overfitting of the training set (B. Yang *et al.*, 2022). In addition, having a sufficient amount of activity data also impacts training of the supervised methods. What is considered sufficient is a tricky question to answer since it is highly dependent on the complexity of the problem. Although factors such as the type of the machine learning algorithm, the number of classification classes (for classification tasks), and the type and number of input features can affect what is deemed sufficient. Many open-source algorithms for *de novo* drug design focus on the development of new algorithms rather than on specific protein targets. Therefore, the development of platforms which integrate the results from multiple *de novo* drug design algorithms for novel drug targets are urgently needed (Wang *et al.*, 2022). This is because the usefulness and suitability of these algorithms for specific protein targets cannot be rated unless they are explicitly tested. If these platforms are made, then it can potentially act as a starting point to guide researchers and pharmaceutical industries when selecting the appropriate *de novo* drug design methods that are suitable for their purpose.

1.10 Project Scope

This project focuses on the development of novel *de novo* RL-based drug design methods in both ligand-based and structure-based scenarios. In particular, I aim to show how the *de novo* drug design methods formulated using reaction-based action spaces I developed results in improvement in molecular properties as compared to the conventional atom/bond-based methods. Figure 1.17 shows the scope of the methods that I have developed using Google's original MolDQN method (Zhou *et al.*, 2019) as inspiration.

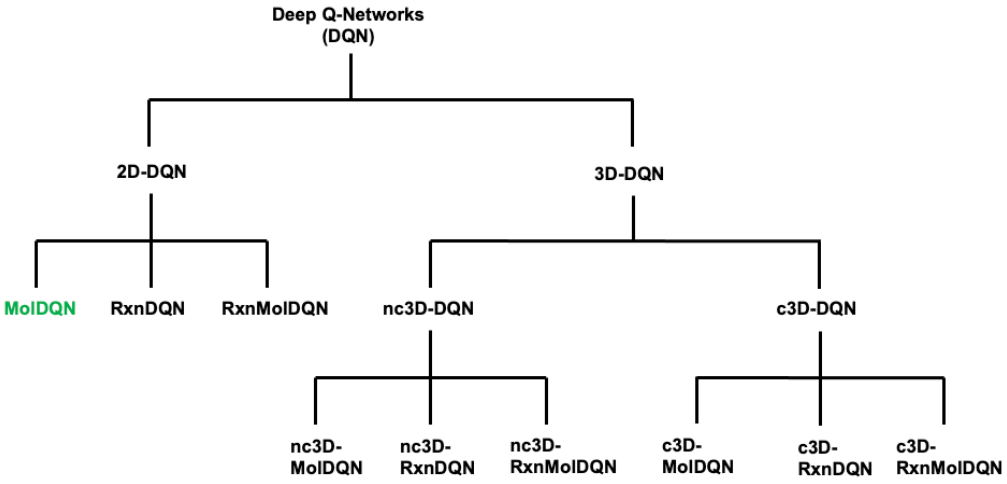


Figure 1.17: Scope of the Deep Q-Networks methods developed during the project. MolDQN (green) is the original method by Zhou *et al.* (2019). RxnDQN and RxnMolDQN are reaction-based methods described in Chapter 2 involving ligand-based *de novo* drug design methods (two-dimensional Deep Q-Networks (2D-DQN)). There are two categories of structure-based methods (or three-dimensional Deep Q-Networks (3D-DQN)) including non-constrained three-dimensional Deep Q-Networks (nc3D-DQN) and constrained three-dimensional Deep Q-Networks (c3D-DQN). Each of these sub-classes in 3D-DQN have MolDQN, RxnDQN and RxnMolDQN variants.

For ligand-based *de novo* drug design (*i.e.*, 2D-DQN), I have developed two new reaction-based methods, consisting of Reaction Deep Q-Networks (RxnDQN) and Reaction-Molecule Deep Q-Networks (RxnMolDQN), which I describe in Chapter 2 by using the reward function of QED and penalized logP to allow for direct comparison with MolDQN. For structure-based *de novo*

drug design, I have developed the non-constrained three-dimensional Deep Q-Networks (nc3D-DQN) by integrating AutoDock Vina (Trott and Olson, 2010) and using predicted ligand efficiency as the reward function, which I introduce in Chapter 3. I also introduced a variant of nc3D-DQN methods by using pharmacophoric-based reward function based on normalized SuCOS (Leung *et al.*, 2019; Leung and Moesser, 2021) to capture the desired shape and pharmacophoric features of a desired active ligand.

SuCOS is based on Combined Overlap Score (COS) (Malhotra and Karanicolas, 2017), which is composed of half chemical feature overlap and half shape overlap (Leung *et al.*, 2019). This method is an open-source alternative method to COS and utilizes two RDKit functions, *ScoreFeats* for chemical feature overlap and *ShapeProtrudeDist* for shape overlap (Equation 1.15).

$$SuCOS = 0.5(ScoreFeats) + 0.5(1 - ShapeProtrudeDist) \quad (1.15)$$

(1 – *ShapeProtrudeDist*) calculates the proportion of the reference molecule's volume that is covered by the query molecule with the option *allowReordering* set to *False*. Since this is an asymmetric shape overlap metric, the score for (1 – *ShapeProtrudeDist*) will be 1 if the query molecule completely covers the reference molecule in volume. Therefore, a score of 1 will be returned even if the reference molecule is larger than the query molecule. *ShapeProtrudeDist* computes the shape protrusion between the two molecules based on their pre-defined alignment and uses a grid to encode the molecular shapes. The score from (1 – *ShapeProtrudeDist*) ranges from 0 (no volume overlap) to 1 (complete volume overlap).

SuCOS's chemical feature overlap score (*i.e.*, *ScoreFeats*) is calculated using RDKit's pharmacophoric feature types (*i.e.*, *Donor*, *Acceptor*, *NegIonizable*, *PosIonizable*, *ZnBinder*,

Aromatic, *Hydrophobe*, and *Lumped Hydrophobe*). For each pharmacophoric feature type, RDKit has a list of SMILES Arbitrary Target Specification (SMARTS) patterns defined in the *BaseFeatures.fdef* file to identify the molecule's chemical features. "Feature map" for the reference molecule is created using the *FeatMap* function. In SuCOS, the default settings with a Gaussian potential centered on each features with a sigma factor of 1 Å is used and only counts overlaps within 2.5 Å. The *ScoreFeats* function scores the pharmacophoric features of the query molecule's set against the reference molecule's feature map. The score is normalized by the number of features in the smaller molecule, which makes it an asymmetric score. In normalized SuCOS (Leung and Moesser, 2021), the feature maps are scored using *Best*. In this *Best* mode, each feature in the query molecule is compared with every feature in the reference feature map. The highest score found is used to score the feature map. Therefore, normalized SuCOS is asymmetric since both the components of shape and chemical features are asymmetric. This feature provides the advantage for normalized SuCOS since the score only depends on how well the query molecule overlaps with the reference molecule and it is independent of any size differences between these two molecules. This unique feature of normalized SuCOS makes it a suitable pharmacophoric-based reward function for DQN-based methods since the query molecule (*i.e.*, generated molecule) may completely deviate from the reference molecule.

As an extension to the nc3D-DQN methods, I have also developed constrained three-dimensional Deep Q-Networks (c3D-DQN), which are described in Chapter 4. In Chapter 5, I introduced methods of evaluation to assesss DQN methods described in Chapters 2 to 4 (methods from Figure 1.17). In Chapter 6, "Results and Discussion" for 15 metrics (Table 5.1) are presented for each method in Figure 1.17 whereever relevant. I will show that my new methods improve upon the

original MolDQN in a wide range of metrics and considerations important to real-world drug discovery.

The code and data from this project will be made available on GitHub (https://github.com/angoto) when I submit the preprints, which is currently under preparation.

2 Exploration of Novel Action Spaces in Deep Q-Networks

2.1 Overview

In this chapter, I will be focusing on the theory and methods that led to the development of novel action spaces in Deep Q-Networks by incorporating a novel reaction-based action space. The hypothesis is that this would allow for a pragmatic exploration of chemical space and generate compounds that are more synthetically accessible. I present two new reaction-based *de novo* design methods: Reaction Deep Q-Networks (RxnDQN, Section 2.3.2) and Reaction-Molecule Deep Q-Networks (RxnMolDQN, Section 2.3.3). Together with Zhou *et al.*'s original method of Molecule Deep Q-Networks (MolDQN, Section 2.3.1) (Zhou *et al.*, 2019), these variants constitute two-dimensional Deep Q-Networks (2D-DQN) methods.

2.2 Background

2.2.1 Markov Decision Process (MDP)

The environment in reinforcement learning (Sections 1.6 and 1.7) is based on Markov Decision Process (MDP). MDP operates on a set of states called the "state space", S; these can be reached by a set of actions that constitute an "action space", A; the probability of taking an action, a, to move from state s to state s' is defined by P_a; and the expected reward of taking this action is $R_a(s, s')$ (Bertsekas, 2007; Kaelbling *et al.*, 1996). The agent may also use a discount factor, $\gamma \in [0, 1)$, which modifies rewards at later times, t, by a factor of γ^t. The goal is to maximize the expected reward of the agent by finding an optimal policy, $\pi^*(s)$, which determines which action the agent should take at each state, s. This policy can be either deterministic or probabilistic, but I will only be considering deterministic policies (Section 1.7) in this thesis. The optimal policy,

$\pi^*(s)$, is found by solving value or policy iterations when the four parameters are defined (*i.e.*, (S, A, P_a, R_a)).

In value iteration, we can solve for $V_t^*(s)$ to find the optimal value function at state s with t time steps remaining by working backwards. If the agent has only one step remaining, then it will take the action with the highest reward as described in Equation 2.1:

$$V_1^*(s) = \max_a R(s,a) \tag{2.1}$$

At a given iteration, t, the agent should take the action where the immediate reward plus the expected discounted value of the next state (*i.e.*, $t+1$) is maximized, as shown in Equation 2.2:

$$V_{t+1}^*(s) = \max_a \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)\, V_t^*(s')\right] \tag{2.2}$$

where $p(s'|s,a)$ represents the state transition probability of moving from state s to state s'. The chain of computations of value iteration is illustrated in Equation 2.3:

$$V_1^* \rightarrow V_2^* \rightarrow V_3^* \rightarrow \cdots \tag{2.3}$$

There are two policies that can be adopted in value iteration to solve for $\pi^*(s)$: finite horizon and infinite horizon. Finite horizon is where the optimization ends after T time steps, so we can stop iterating after computing for V_T^*. We can find $\pi_{t+1}^*(s)$ (which corresponds to the optimal action to take at state s at iteration $t+1$) by finding the argmax of $V_{t+1}^*(s)$ (Equation 2.4), instead of the max formula in Equation 2.2:

$$\pi_{t+1}^*(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)\, V_t^*(s')\right] \tag{2.4}$$

For infinite horizon, the optimal V^* can be thought of as the limit of the above sequence in Equation 2.3 where $V^* = \lim_{t \to \infty} V_t^*$. We cannot reach V^* in a finite number of iterations, so we pick some convergence tolerance, θ, and stop iterating once the value at each state after an update change is less than θ, where $|V_{t+1}^*(s) - V_t^*(s)| \leq \theta$ for all states s. Even though V_t^* does not

reach V^* in finite time, the policy created using V_t^* converges to π^* after a finite number of iterations. By setting a low enough θ, we use the resulting value function of V to achieve $\pi^*(s)$ as shown in Equation 2.5:

$$\pi^*(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)\, V(s')\right] \tag{2.5}$$

In the case of infinite horizon (Equation 2.5), the policy does not depend on time steps, unlike in the case of finite horizon (Equation 2.4).

The optimal policy, $\pi^*(s)$, can also be found through policy iteration. In policy iteration, a two-step process is repeated. The chain of computations in policy iteration is described in Equation 2.6:

$$\pi_1(s) \rightarrow V^{\pi_1}(s) \rightarrow \pi_2(s) \rightarrow V^{\pi_2}(s) \rightarrow \cdots \tag{2.6}$$

where $\pi_1(s)$ is some initial policy used to create the corresponding value function of $V^{\pi_1}(s)$ and this value function is used to create an updated policy of $\pi_2(s)$ Therefore, the first procedure in policy iteration computes $V^{\pi_t}(s)$ from $\pi_t(s)$, which is known as policy evaluation. $V^{\pi_t}(s)$ informs the expected value gained by following policy of π_t starting from state s. Policy evaluation can be conducted iteratively in a similar way to the value iteration as defined in Equation 2.1, but this time by removing max over a and replacing a with the policy action of $\pi_t(s)$ as shown in Equation 2.7:

$$V^{\pi_t}(s) = R(s, \pi_t(s)) \tag{2.7}$$

In the second procedure in policy iteration, $\pi_{t+1}(s)$ is found using the value function $V^{\pi_t}(s)$ by taking the action that maximizes the immediate reward plus the expected future reward as shown in Equation 2.8:

$$\pi_{t+1}(s) = \arg\max_a \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)\, V^{\pi_t}(s')\right] \tag{2.8}$$

In finite time, the policy will converge and reach some optimum π^*. Once this occurs, the update step will no longer change the policy. Since π_t is used to compute V^{π_t}, this implies that the value function also converges and will not receive further updates.

The preference for value iteration or policy iteration depends on the type of MDP, but policy iteration tends to be a more preferred way since it takes fewer iterations to find π^*. In terms of computational time, policy iteration takes longer per iteration than value iteration since extra time is needed to conduct the policy evaluation step. It is also worth noting that the discount factor, γ, can drastically impact the learned policy and value function. When γ is small, the rewards that occur in the immediate future are very heavily weighted as compared to the rewards that might happen later. In this case, the agent is likely to be very short-sighted and optimizes around the first few runs. When larger γ is used, the agent is not as concerned about the immediate reward, so it is willing to incur penalties in the short-term if it leads to even larger rewards in the long-term. Infinite horizon always has a γ, but this is optional for finite horizon.

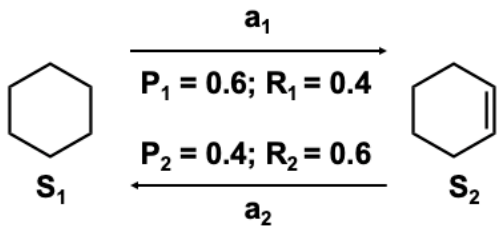


Figure 2.1: A simple example of a Markov Decision Process (MDP). By taking action a_1, (here, the addition of a bond) the system moves from state S_1 to state S_2, and this action is performed with a probability of 0.6 and resultant reward of 0.4. P_x and R_x represent probability and reward, respectively, where x is either 1 or 2.

A simple example of a Markov Decision Process is shown in Figure 2.1 where S and a represent the states and actions, respectively. As shown in Figure 2.1, a key property of MDP is the ability

to transfer from one state to another and back again by taking forward and reverse actions. In the case of MolDQN, these forward and reverse actions consist of the addition of atoms or bonds, and removal of bonds that would only remove one atom or decrease the bond order, respectively (Zhou *et al.*, 2019). In Section 2.3, I will describe new action spaces in Deep Q-Learning (Section 2.2.4) for *de novo* design. In RxnDQN (Section 2.3.2), I introduce new forward and reverse actions which consist of forward chemical reactions and their corresponding retrosynthetic disconnection reactions, respectively.

2.2.2 Reinforcement Learning (RL)

RL also uses an MDP (Section 2.2.1), but the task is to learn the optimal policy, $\pi^*(s)$. This is no longer achieved through policy iteration or value iteration since these depend on transition probabilities P and rewards R. Therefore, RL needs to find a way of moving around the state space and learning the environment so it can find the optimal π^*. Model-based learning and model-free learning can be used to achieve this (Moerland *et al.*, 2020; Sutton and Barto, 2018).

In model-based learning, we estimate P and R by transitioning around the state space and recording the observations. A record of the reward and the state transition is made whenever an action is taken to transition within a state space. By recording state s' by taking an action a many times, it is possible to estimate $p(s'|s,a)$ since it considers the frequency of times that state s' was transitioned from state s. The same analogy applies for estimating R. Once there is an estimate of P and R, it is possible to use planning methods such as policy iteration or value iteration (Section 2.2.1) to solve for $\pi^*(s)$.

One of the disadvantages of model-based learning is that it is computationally expensive to keep track of all the rewards and state transition probabilities (Ruiz-Montiel *et al.*, 2017). In addition, there may be a need to make many visits to each state by taking the same action multiple times to get a good estimate of P. This process requires a lot of exploration to be performed. Therefore, model-based learning requires a lot more work than is necessary just to optimize for π^*. However, the benefit of using model-based learning is when P and R change over time (Sutton and Barto, 2018). In this case, we can just update the corresponding entries in the model and revise the planning process to the new system to get the new π^*. This is not possible without a model for P or R and there is a need for extra time to learn the effects caused by these changes.

Model-free learning is an alternative way to move around the state space and learn the environment. In model-free learning, we do not explicitly keep track of P and R as in model-based learning. Instead, we try to learn a Q-function, where Q stands for "quality" and represents the quality of state-action pair, (s, a). The Q-function, $Q(s, a)$, defines the long-term value for taking an action a at state s. The optimal Q-function, Q^*, obeys the Bellman equation as shown in Equation 2.9:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) \max_{a'} Q^*(s',a') \tag{2.9}$$

where the first term denotes the immediate reward gained by performing the action a and the second term is the expected reward by taking the optimal follow-up action a'. By using Equation 2.9, it is possible to find $\pi^*(s)$ by considering which action a maximizes the Q-function at state s as shown in Equation 2.10:

$$\pi^*(s) = \arg\max_{a} Q^*(s,a) \tag{2.10}$$

If P and R are known, then Q^* can be easily computed using planning methods (Section 2.2.1), which can then be used to compute π^*. However, in model-free learning, the parameters P and R are unknown, so we need to estimate Q^* by transitioning around the state space. An estimate of $Q(s,a)$ is initially set to arbitrary values at the beginning of the task. Each time an action a at state s is taken, we update the Q-function, $Q(s,a)$, based on the observation. The goal for model-free learning is to update $Q(s,a)$ such that $Q(s,a) \rightarrow Q^*(s,a)$ as actions are taken in the state space. There are two main ways to update $Q(s,a)$: Q-learning and SARSA (State Action Reward State Action) (Sutton and Barto, 2018; Watkins and Dayan, 1992).

Since a policy π (Equation 2.10) defined above informs how to move around the state space, we need a way to update the initial Q-values based on the observations made when transitioning around the state space. $Q(s,a)$ can be updated by supposing that we are at state s and after taking an action a, we get reward r by moving to state s' as shown in Equation 2.11:

$$Q(s,a) \leftarrow Q(s,a) + \alpha_t[\Delta Q] \tag{2.11}$$

where α_t is the learning rate at time-step, t, and ΔQ is determined by the update method that is used. The full update rules for Q-learning and SARSA are shown in Equations 2.12 and 2.13, respectively.

$$Q(s,a) \leftarrow Q(s,a) + \alpha_t[R(s,a) + \gamma Q(s',a^*) - Q(s,a)] \tag{2.12}$$

$$Q(s,a) \leftarrow Q(s,a) + \alpha_t[R(s,a) + \gamma Q(s',a') - Q(s,a)] \tag{2.13}$$

where a' is the action taken at state s' and a^* is the action with the highest Q-value at state s'.

Q-learning updates the Q-function, $Q(s,a)$, using $R(s,a) + \gamma Q(s',a^*)$ as shown in Equation 2.12. The Q-function is updated based on the immediate reward obtained from (s,a) plus the quality of the greedy follow-up action a^* at the next state s'. An ε-greedy policy determines the

action that the agent takes at state s' (Section 2.3.1). Q-learning is an "off-policy" technique, but this term does not mean that Q-learning does not follow a policy (Metz *et al.*, 2017). It is called an "off-policy" technique since the update of the Q-function does not use action a' given by the policy. Instead, the updates are based on the action a^* with the highest Q-value. In the case of SARSA, the Q-function is updated by using $R(s,a) + \gamma Q(s',a')$. Therefore, SARSA is an "on-policy" technique since the update of the Q-function is based on the reward from the transition plus the quality of the actually performed follow-up action a' at the next state s' (Sutton and Barto, 2018). If the ε-greedy policy sets the follow-up move of a' to be the action with the largest Q-value (*i.e.*, the so-called "greedy" option), then the updates performed by Q-learning and SARSA are identical. However, if the policy conducts some random action a', then SARSA updates the Q-function by taking the random action a' as a follow-up, whereas Q-learning takes the greedy action as a follow-up.

2.2.3 Deep Q-Networks (DQN)

Q-learning discussed in Section 2.2.2 encounters a problem with memory and high computational requirements if the combination of states and actions are too numerous. To combat this issue, in 2015, DeepMind introduced a deep reinforcement learning algorithm that was able to play Atari 2600 video games known as "Deep Q-Networks" (DQN) (Mnih *et al.*, 2015). DQN was applied to *de novo* molecular design to learn the optimal policy, π^*, in MolDQN (Zhou *et al.*, 2019) and in other methods introduced in Chapter 1 (Figure 1.17). In Q-learning, a state s is associated with an action a – leading to one Q-value per state-action pair. In DQN, a state s (*i.e.*, single input) can give as many Q-values as possible actions, a. In DQN, the action that leads to a highest Q-value will be chosen as the action to take from the current state s.

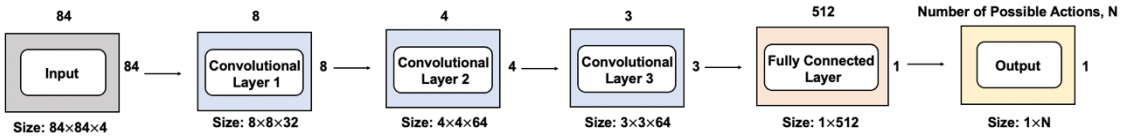


Figure 2.2: Architecture of DQN (left to right): Input, Convolutional Layers, Fully-connected Layer, and Output (Mnih *et al.*, 2015).

The architecture of DQN (Figure 2.2) used to learn how to play the video games consisted of three convolutional and one fully-connected hidden layers. The term "hidden" layer is used for any layer that is not an input or output. The input to this neural network was a stack of four Atari 2600 frames (size: 84×84×4) that were preprocessed to reduce computational time in the training. The preprocessing of these frames involved converting RGB (Red-Green-Blue) to grayscale and skipping non-useful areas. Several images are required to have information about motion and in particular, where and how fast objects are moving. The first three convolutional layers (Figure 2.2; 8×8×32, 4×4×64, and 3×3×64) are used to exploit spatial relationships in images and spatial properties across frames of the stack. The nodes in these convolutional layers employ an activation function called a Rectifier Linear Unit (ReLU), which has been shown to be effective for convolutional layers (Agarap, 2018). The role of the fully-connected layers (1×512 and the output layer 1×N where N is the number of possible actions) is to estimate the Q-value for each action from the outputs of the previous layer. Therefore, this architecture provides a vector of Q-values from a state s. The highest Q-value then defines the best action in DQN, just like Q-learning.

DQN is a type of convolutional neural network where the Bellman equation (Mnih *et al.*, 2015) is used as an iterative update to minimize the loss function shown in Equation 2.14.

$$L(\theta) = \mathbb{E}[(R(s,a,s') + \gamma \max_{a'}\big(Q(s',a',\theta)\big) - Q(s,a,\theta))^2] \quad (2.14)$$

where θ is the parameters of the network, $R(s,a,s') + \gamma \max_{a'}\big(Q(s',a',\theta)\big)$ is the target, and $Q(s,a,\theta)$ is the prediction (remaining parameters are defined in Sections 2.2.1-2.2.2). In DQN, the actions resulting from first iterations are often bad, but it improves progressively as the network learns and update its parameters. However, it is difficult to use deep learning with reinforcement learning (*i.e.*, deep reinforcement learning). This is due to the instability of DQN since the parameters of Q-network are always updated to obtain a better estimation of Q-values and when playing dynamic video games, the images given as input are correlated and evolve progressively. One of the solutions to this issue is using *prioritized experience replay* (Schaul *et al.*, 2015), such as in the case of MolDQN. In prioritized experience reply, the input video game frames to the neural network are placed into a buffer and a batch of samples (size 32) is taken to train the neural network. Since some of these batches are more important than others, these batches would need to be sorted by a criterion that defines the priority of each experience. This preference value is computed based on the difference between the prediction and the target Q-values. If the error is higher, then it implies that a particular experience has something more to teach to the network.

2.2.4 Deep Q-Learning

In deep Q-learning, the goal is to find an estimate of the Q-function (Section 2.2.2) by learning the optimal policy, π^*, that selects the best action to take from the current state that would maximize the resultant reward (Zhou *et al.*, 2019). The notation used in this section are as in Sections 2.2.1-2.2.3 unless otherwise stated.

Q-learning (Section 2.2.2) in MolDQN (Zhou *et al.*, 2019) operates on a molecule, m, and maximizes the reward by taking an action on the molecule at each step, t, that is defined by the Q-function as shown in Equation 2.15:

$$Q^{\pi}(s,a) = Q^{\pi}(m,t,a) = E_{\pi}\left[\sum_{n=t}^{T} r_n\right] \tag{2.15}$$

where π is the policy, a is the action at state s, E_{π} is an expectation with respect to π, r_n is the reward at step n, m is the molecule at step t, and T is the maximum number of steps. This Q-function calculates the future rewards that taking an action a on state s would have, and the subsequent actions are determined by the policy, π. Therefore, the optimal policy is defined in Equation 2.16 as:

$$\pi^*(s) = \arg\max_{a} Q^{\pi^*}(s,a) \tag{2.16}$$

where π^* is the optimal policy and a is the action at state s. It is possible to find $\pi^*(s)$ by considering which action a maximizes the Q-function. In the case of MolDQN, since a deterministic MDP (Section 1.7) and an accurate model of the environment is used, the value function, V, is defined in Equation 2.17 as:

$$V(s) = \max_{a} Q(s,a) \tag{2.17}$$

where the agent will take the action a with the highest Q-function. Therefore, the Q-function in MolDQN is calculated using Equation 2.18:

$$Q(s,a) = R(s') + V(s') \tag{2.18}$$

when $R(s')$ is the expected reward that would be obtained by moving from state s to state s' by taking action a, and $V(s')$ is the corresponding value function. Since the number of steps in MolDQN is limited, the MDP (Section 2.1.1) is time-dependent and the optimal policy, π^*, will be time-dependent as well. The Q-function in DQN is parameterized as $Q(s,a,\theta)$ where θ effectively corresponds to the parameters of the deep neural network, such as the weights and

biases. This is referred to as a neural network function approximator in deep Q-learning. This approximator gets trained by minimizing the loss function as shown in Equation 2.19:

$$L(\theta) = \mathbb{E}\left[f_1\left(y_t - Q(s_t, a_t; \theta)\right)\right] \tag{2.19}$$

where t is the current iteration, $y_t = r_t + \max_a Q(s_t, a; \theta)$ is the target value, and f_1 is a loss function.

The goal of the loss function is to minimize the difference between the target value and the predicted value, $Q(s_t, a_t; \theta)$, and is known as TD error. In MolDQN, the Huber loss function is used as shown in Equation 2.20 (Boyd and Vandenberghe, 2004):

$$f_1(x) = \begin{cases} \frac{1}{2}x^2 & if\ |x| < 1 \\ |x| - \frac{1}{2} & otherwise \end{cases} \tag{2.20}$$

This loss function is quadratic for small values of x and linear for larger values, with equal values and slopes of the component functions when $|x| = 1$.

Q-learning retrains the Q-function at every episode in order to select the best action at each state. In the case of MolDQN, double Q-learning is used to enable different policies (*i.e.*, two Q-functions from Equation 2.18) to be applied to select the next action at each state and estimate the Q-function (M. Clark *et al.*, 2006). By assigning experience randomly to train one of two Q-functions, the algorithm trains two sets of parameters, θ and θ'. When updating the Q-function, one set of parameters is used to select the action and the other is used to evaluate the action. Therefore, double Q-learning stores two Q-functions, Q^A and Q^B where each Q-function is updated with a value from the other Q-function to select the next action. This enables double Q-learning to overcome any errors that might be exhibited by relying on a single Q-function (van Hasselt *et al.*, 2016).

2.2.5 Action Space

The action space is defined as the set of all possible actions that an agent could take to maximize the reward (Bertsekas, 2007). In the case of MolDQN (Section 2.3.1), the action space consists of the addition of atoms or bonds, and removal of bonds. Hence, throughout this thesis, I will refer to MolDQN as having an "atom/bond-based action space".

One of the innovations I introduced was the use of reactions and retrosynthetic disconnections instead of atom/bond-based actions. This method is referred to as "RxnDQN". For RxnDQN (Section 2.3.2), the new action space is based on the top 10 most commonly-used chemical reactions in medicinal chemistry (Roughley and Jordan, 2011). RxnDQN thus has "reaction-based action space".

In the case of RxnMolDQN (Section 2.3.3), since it combines the action spaces of MolDQN and RxnDQN, I will refer to it as having a "hybrid action space".

2.2.6 Multi-objective Optimization in Reinforcement Learning

Multi-objective optimization in reinforcement learning is essentially an extension of single objective optimization, but multiple criteria are simultaneously optimized to reach an optimal reward by applying a reward function that amalgamates multiple objectives into a single score. This type of optimization is particularly useful in drug discovery since it is itself a multi-objective optimization problem (J. P. Hughes *et al.*, 2011). A successful drug must be potent, bioavailable, safe, and relatively easy to synthesize in bulk, *etc.*

In multi-objective reinforcement learning, the environment will return a vector of rewards at each step, t, with one reward for each objective as shown in Equation 2.21:

$$\overrightarrow{r_t} = \left[\overrightarrow{r_{1,t}}, \ldots, \overrightarrow{r_{k,t}}\right]^T \in \mathbb{R}^k \tag{2.21}$$

where k is the number of objectives and T is the maximum number of steps. In MolDQN, a "scalarized" reward framework is implemented in multi-objective optimization by introducing a user-defined weight vector, $w = [w_1, w_2, \ldots, w_k]^T \in \mathbb{R}^k$. In this way, the scalarized reward can be computed using Equation 2.22:

$$r_{s,t} = w^T \overrightarrow{r_t} = \sum_{i=1}^{k} w_i r_{i,t} \tag{2.22}$$

In the case of 2D-DQN methods (Figure 1.17), I investigated multi-objective optimization by using a combination of QED and Tanimoto similarity between the *de novo* designed compound and a desired target compound (see Section 2.3.1, and the results in Section 6.18.1).

2.2.7 RDKit

The DQN methods described in this work were developed using RDKit and TensorFlow (Section 2.2.8). RDKit is an open-source cheminformatics toolkit (Berkeley Source Distribution (BSD) license) written in C++, but which can be used from Python or Java (Landrum, 2019). It consists of a collection of useful functionalities for manipulating small molecules, their chemical structures, performing virtual chemical reactions, generating coordinates (2D and 3D), and computing fingerprints, *etc.*

2.2.8 TensorFlow

TensorFlow is an open-source Application Programming Interface (API) for machine learning and artificial intelligence applications (Apache License 2.0), which can be used in Python, JavaScript, C++, and Java (Abadi *et al.*, 2015). This library is used for training and inference of deep neural networks.

2.2.9 Reaction Cheminformatics

The reaction-based action space used in this work is inspired by the poised fragment-based approach of Cox *et al.* (2016), who used reaction SMARTS to build virtual libraries using XPoise (fragment elaboration software). XPoise was built using RDKit (Landrum, 2019) and KNIME (Berthold *et al.*, 2009), where reactions encoded using reaction SMARTS strings were used to conduct the fragment elaboration using building blocks libraries from vendor catalogues (Cox *et al.*, 2016). When SMARTS are used in reactions queries, it is commonly referred to as "reaction SMARTS".

SMARTS is a cheminformatics query language developed by Weininger, *et al.*, and the details of the language can be found in the Daylight SMARTS theory manual (Daylight-Chemical-Information-Systems, 1997). All SMILES strings are also valid SMARTS strings, but the semantics changes because SMILES describes molecules, whereas SMARTS describes patterns. The molecule represented by a SMILES string is usually (but not always) matched by the same string in SMARTS. For example, the SMILES of "C1=CC=CC=C1" (cyclohexatriene) is interpreted as benzene. This molecule will be matched by the SMARTS of "c1ccccc1", which is interpreted as the pattern of 6 aromatic carbons in a ring. The SMARTS of "C1=CC=CC=C1"

makes a pattern of six aliphatic carbons in a ring with alternating single and double bonds. Therefore, the SMARTS expression of "C1=CC=CC=C1" does not match benzene, but it will match the non-aromatic phenyl cation with SMILES of "C1=CC=CC=[C+]1". In addition, when atoms are specified without brackets in SMILES, default values are used, but for SMARTS, unspecified properties are not defined to be part of the pattern. For example, SMILES of "O" means an aliphatic oxygen with zero charge and two hydrogens (*i.e.*, water). Although in SMARTS, the same expression means any aliphatic oxygen regardless of charge, hydrogen count, *etc.* and so it will match the oxygen in water, but also in ethanol, acetone, molecular oxygen, *etc.* If [OH2] is specified in SMARTS, then it limits the pattern to match only water, which is also the fully specified SMILES for water. Therefore, the preference for SMARTS and SMILES depends on the usage and what the user is trying to achieve when using these expressions. However, the SMARTS expression is considered as an extension of SMILES that enables specification of chemical patterns.

In reaction SMARTS, atoms are described either using their atomic number (*e.g.*, "[#6]" for carbon) or by their chemical symbol (*e.g.*, "c" or "C" for carbon). For chemical symbols as in SMILES strings, capital letters represent aliphatic atoms and lower-case letters denote aromaticity. There are other useful symbols to describe chemical properties such as chirality ("@" and "@@"), valence ("v"), wildcard atoms ("*"), and hydrogen count ("H"). Complex queries are possible using the four logical operators: "," (or); "!" (not); "&" (high priority and); ";" (low priority and).

A reaction between reactants A and B to generate product C can be expressed in the form "A.B >> C" using reaction SMARTS as noted in Appendix A, Table A2. Product and reactant substructures are encoded in separate reaction SMARTS and only the reaction center is defined.

After mapping the persistent atoms in reaction SMARTS, any unmapped atom gets removed from the reactant and added to the product. This includes unmapped atoms common to reactant and product. In mapping, ":integer" is added to an atom (*e.g.*, [#6:1]). It is important to map atoms during this process since we may run into a risk of losing stereoinformation. R-groups that are undefined remain unchanged after performing the reaction.

In reaction-based DQN methods, these reaction SMARTS in Appendix A, Tables A1-A2 are implemented using RDKit to define the action space of MDP.

2.2.10 Penalized logP & Synthetic Accessibility Score (SAscore)

Penalized logP is one of the single objective optimization reward functions originally used in MolDQN, which has been obtained for comparison in my new 2D-DQN methods (Zhou *et al.*, 2019). LogP is the base-10 logarithm of the partition coefficient of a compound between octanol and water. Penalized logP is computed using Equation 2.23:

$$Penalized\ logP = logP - SAscore - \ Number\ of\ Long\ Cycle \tag{2.23}$$

where "SAscore" is the synthetic accessibility score (Ertl and Schuffenhauer, 2009) and "Number of Long Cycle" counts the number of rings that have more than six atoms. SAscore is a predicted metric that considers how difficult it is to synthesize a compound.

SAscore is calculated by subtracting *complexityPenalty* from *fragmentScore* as shown in Equation 2.24:

$$SAscore = fragmentScore - complexityPenalty \tag{2.24}$$

The *fragmentScore* aims to capture the "historical synthetic knowledge" by calculating the sum

of contributions of all fragments (defined by the PubChem collection) in the molecule divided by the number of fragments in the molecule. The database for *fragmentScore* was generated by conducting a statistical analysis of 605,864 different fragment types, which have been obtained by fragmenting 934,046 representative molecules from the PubChem collection (Kim *et al.*, 2021). The most common fragments are shown in Figure 2.3.

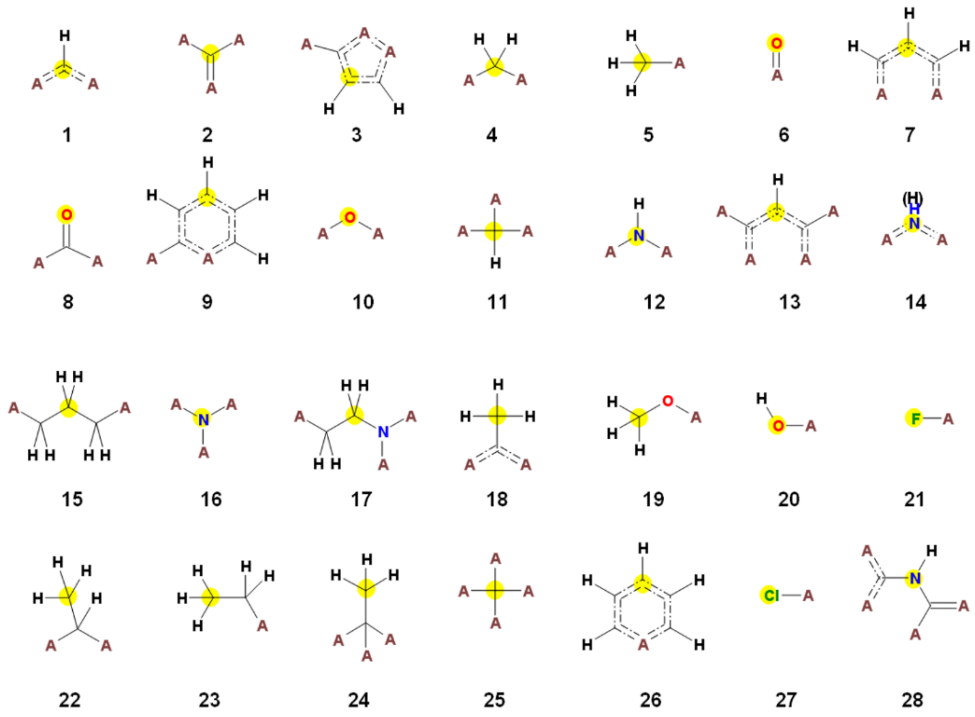


Figure 2.3: Example of most common fragments present in a subset of PubChem molecules. "A" represents any non-hydrogen atom, dashed double bond indicates an aromatic bond, and the yellow circle is the central atom of the fragment. This figure is reproduced from original author (licensed under Creative Commons CC BY) (Ertl and Schuffenhauer, 2009).

The *complexityPenalty* characterizes the complex structural features in the molecule, which is calculated from a combination of the *ringComplexityScore* (Equation 2.25), *stereoComplexityScore* (Equation 2.26), *macrocyclePenalty* (Equation 2.27), and *sizePenalty* (Equation 2.28).

$$ringComplexityScore = \log(\mathrm{n}RingBridgeAtoms + 1) + \log(\mathrm{n}SpiroAtoms + 1) \quad (2.25)$$

$$stereoComplexityScore = \log(\mathrm{n}SteroCenters + 1) \quad (2.26)$$

$$macrocyclePenalty = \log(\mathrm{n}Macrocycles + 1) \quad (2.27)$$

$$sizePenalty = \mathrm{n}Atoms^{1.005} - \mathrm{n}Atoms \quad (2.28)$$

The *ringComplexityScore* determines the complexity of a molecule with ring systems based on spiro rings and ring fusions. The *stereoComplexityScore* penalizes molecules with multiple stereocenters. The *macrocyclePenalty* increases when the ring size is greater than eight in a molecule. The *sizePenalty* penalizes molecules with more atoms. After subtracting c*omplexityPenalty* from *fragmentScore* (normally in the range of -4 (worst) to 2.5 (best)), the value is multiplied by -1 and scaled to be in between 1 to 10. This scaling is conducted simply for ease of interpretation. The molecules with high SAscore (closer to 10) are more difficult to synthesize than the molecules with low SAscore (closer to 1).

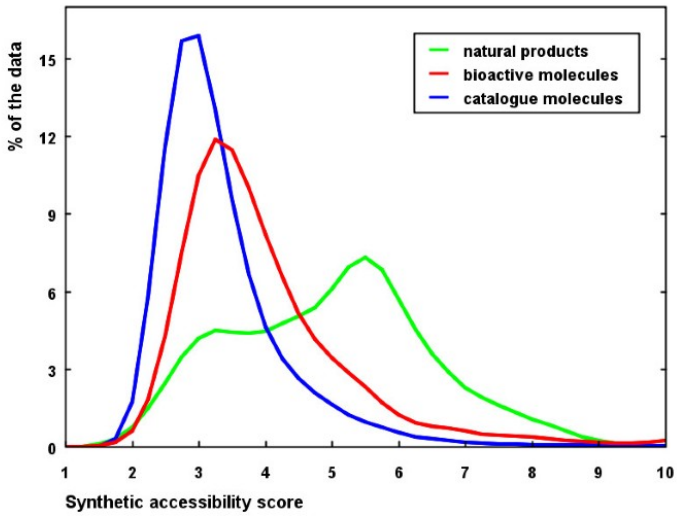


Figure 2.4: Distribution of Synthetic Accessibility Score (SAscore) for Natural Products (green), Bioactive Molecules (red), and Catalogue Molecules (blue). SAscore ranges between 1 (most synthetically accessible) to 10 (least synthetically accessible). This figure is reproduced from original author (licensed under Creative Commons CC BY) (Ertl and Schuffenhauer, 2009).

Typically, catalogue molecules have an SAscore, of 2 to 4, while natural products range from 2 to 7 as shown in Figure 2.4 (Ertl and Schuffenhauer, 2009). Therefore, the value of penalized logP ranges from -∞ (less favorable) to +∞ (more favorable).

2.2.11 Quantitative Estimate of Druglikeness (QED)

QED is another reward function considered for the 2D-DQN methods, which is a measure of druglikeness based on the concept of desirability as shown in Equation 2.29 (Bickerton *et al.*, 2012):

$$QED = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \ln d_i\right) \tag{2.29}$$

where d is a series of desirability functions where each corresponds to a different molecular descriptor and n corresponds to the number of descriptors. This is the empirical rationale of QED where it reflects the underlying distribution of molecular properties (d): molecular weight (d_{MW}), octanol-water partition coefficient (AlogP; d_{AlogP}), number of hydrogen bond donors (d_{HBD}), number of hydrogen bond acceptors (d_{HBA}), molecular polar surface area (d_{PSA}), number of rotatable bonds (d_{ROTB}), number of aromatic rings (d_{AROM}), and number of structural alerts (d_{ALERTS}) (*i.e.*, unwanted chemical functionalities). It has been found that these property distributions are consistently best modelled using asymmetric double sigmoidal (ADS) function as shown in Equation 2.30:

$$d(x) = a + \frac{b}{\left[1+\exp\left(-\frac{x-c+\frac{d}{2}}{e}\right)\right]}\left[1 - \frac{1}{\left[1+\exp\left(-\frac{x-c+\frac{d}{2}}{f}\right)\right]}\right] \tag{2.30}$$

where $d(x)$ is the desirability function for molecular descriptor, x. The parameters (a, b, c, d, e, and f) are used for each of the ADS functions (d_{MW}, d_{AlogP}, d_{HBD}, d_{HBA}, d_{PSA}, d_{ROTB}, d_{AROM}, d_{ALERTS}). The details of the parameters can be found in the original work (Bickerton *et al.*, 2012).

Since the chosen molecular descriptors (d) vary in the importance of their contribution to druglikeness, each descriptor is weighted by their relative importance. The unweighted QED is shown in Equation 2.29 and the weighted QED (QED_w) is shown in Equation 2.31:

$$QED_w = \exp\left(\frac{\sum_{i=1}^{n} w_i \ln d_i}{\sum_{i=1}^{n} w_i}\right) \tag{2.31}$$

where d is the individual desirability function, w is the weight applied to each function, and n is the number of descriptors. Therefore, the complete weighted QED is given in Equation 2.32 where details of the weight and desirability function for each descriptor can be found in the original work (Bickerton *et al.*, 2012). The score of QED ranges between 0 (least druglike) to 1 (most druglike).

$$QED_w = \exp\left[\frac{A}{B}\right] \tag{2.32}$$

where

$$A = W_{MW} \ln d_w + W_{ALOGP} \ln d_{ALOGP} + W_{HBA} \ln d_{HBA} + W_{HBD} \ln d_{HBD} + W_{PSA} \ln d_{PSA} + W_{ROTB} \ln d_{ROTB} + W_{AROM} \ln d_{AROM} + W_{ALERTS} \ln d_{ALERTS} \tag{2.33}$$

$$B = W_{MW} + W_{ALOGP} + W_{HBA} + W_{HBD} + W_{PSA} + W_{ROTB} + W_{AROM} + W_{ALERTS} \tag{2.34}$$

W_x and d_x are the weight and desirability function of each descriptor, x, respectively.

QED is used in the 2D-DQN methods, but it should be noted that RDKit's implementation of QED (Landrum, 2019), is not completely identical to the original work of Bickerton *et al.* (2012). These differences are caused due to the underlying calculated property calculators used in both methods, such as the discrepancies between the logP calculations (Wildman and Crippen, 1999). However, the resulting differences in QED are very small and do not compromise the usefulness of QED as implemented by RDKit.

2.2.12 Reaxys Database

The Reaxys database consists of chemistry information and data from published literature including journals and patents, which is licensed by Elsevier (Elsevier, 2020). This information includes materials such as chemical reactions, chemical compounds, chemical properties, synthesis planning, and experimental procedures.

In my RxnDQN and RxnMolDQN methods (Sections 2.3.2-2.3.3), reagents from the Reaxys database whose chemical reactions gave greater than 90% yield were used for the reaction-based action space, to help improve synthetic feasibility of the generated compounds. This "greater than 90% yield" is the reported yield in Reaxys for each reaction selected. Reagents from the Reaxys database were retrieved using the Reaxys API in collaboration with Roche (Fitzner, 2019, 2020; Schindler, 2019) using the reaction SMARTS defined in Appendix A, Table A2. The number of the corresponding reagents for each reaction is given in Appendix B, Table B1.

2.2.13 DrugBank Database

The DrugBank database contains information on approved, withdrawn, investigated, and other types of drugs and their targets (where known), and is curated by the University of Alberta and The Metabolomics Innovation Center, Canada (Wishart *et al.*, 2018).

In order to test the performance of 2D-DQN methods, clinically approved drugs from DrugBank were downloaded from the website (Wishart *et al.*, 2018); 100 approved drugs were selected using the procedures described in Section 5.2 and used as target molecules in the various 2D-DQN methods.

2.2.14 Project Aims

As mentioned in Section 1.2, the size of the drug-like chemical space is estimated to be on the order of 10^{60}-10^{100} small molecules (Drew *et al.*, 2012). Therefore, the ability to explore chemical space efficiently is an essential aspect of discovering new drugs quickly. MolDQN (Zhou *et al.*, 2019) uses small changes like adding or removing atoms and/or bonds, which I hypothesized would limit its ability to explore chemical space efficiently. Moreover, the proposed molecules are often not synthetically accessible (or available). I therefore decided to test new methods that could improve MolDQN.

I worked to improve MolDQN by changing the action space to incorporate transformations based on popular synthetic reactions used in medicinal chemistry (Roughley and Jordan, 2011), and developed a new kind of framework called RxnDQN. This allows for compound designs that are more synthetically accessible while at the same time allowing us to address the classic multi-objective optimization problem in drug discovery. The benefit of using this new framework is that we would know exactly which starting material, reagents, and reactions would be required to produce the desired compound. Moreover, this new action space has allowed us to perform a more "global" search of chemical space by enabling chemical modifications that are much larger than adding and removing atoms and bonds. The expansion of the action space by considering the most commonly used reactions in medicinal chemistry in combination with available reagents also implies that we are able to explore chemical space more pragmatically, by generating compounds that are both 100% valid SMILES strings and more likely to be synthesizable than compounds proposed by MolDQN. However, one possible drawback of this new reaction-based action space in RxnDQN is that the coverage of "local" chemical space is limited by the available reagents, so making smaller chemical modifications such as one atom or bond at a time is challenging.

Therefore, I developed a hybridized version that combined both the original atom/bond-based action space of MolDQN with the reaction-based action space of RxnDQN to enable compound generation using both "local" and "global" exploration of chemical space, via my RxnMolDQN.

Table 2.1: Summary of *de novo* methods developed for the 2D-DQN methods, and which action spaces and reactions were used, together with their reward functions. Note, MolDQN acted as the basis of this work (Zhou *et al.*, 2019), while I developed RxnDQN and RxnMolDQN. p.logP and T stand for penalized logP and Tanimoto similarity, respectively.

Method	Atom/Bond Action Space	Reactions Action Space	Single Objective Reward Function	Multi-Objective Reward Function
MolDQN	Yes	No	QED	QED+ T
MolDQN	Yes	No	p.logP	p.logP + T
RxnDQN	No	Yes	QED	QED+ T
RxnDQN	No	Yes	p.logP	p.logP + T
RxnMolDQN	Yes	Yes	QED	QED+ T
RxnMolDQN	Yes	Yes	p.logP	p.logP + T

In this work, I aim to demonstrate the importance of choosing the right action space to guide the reinforcement learning by comparing atom/bond-based action space of MolDQN, reaction-based action space of RxnDQN, and hybridized action space of RxnMolDQN. The "Results and Discussion" for this comparison are shown in Chapter 6. Table 2.1 shows a summary of the various methods I have tested or developed for 2D-DQN methods (two new methods (RxnDQN and RxnMolDQN) where each was tested with four different reward functions (QED, p.logP, QED + T, and p.logP + T)).

2.3 Methodology

2.3.1 Molecule Deep Q-Networks (MolDQN)

MolDQN (Zhou *et al.*, 2019) is written in Python and performs the modification or optimization to a new molecule by using only three actions: atom addition, bond addition, and bond removal. In MolDQN, the environment for RL (Section 2.2.1) is defined using MDP. The differences

between the previous MDP parameters defined in Section 2.2.1 are that R_a is a reward function for scoring the state (m, t), where m is a RDKit-valid molecule (Landrum, 2019) described by a SMILES string, and t is the number of steps taken in this episode (Figure 2.5). For state space S, m could be a specific molecule or an "empty" molecule with no atoms or bonds at the start of the run ($t = 0$). The terminal state is reached for each episode when the optimal value of the reward function is attained in the final step in an episode ($t = T$). A discount factor (Section 2.2.1), γ, is used towards the value of the rewards at a state with step t, which is defined as γ^{T-t}, where γ is typically set to 0.9. This factor ensures that the final state within the maximum number of steps, T, is weighted most heavily. The summary of the parameters in MDP is shown in Figure 2.5.

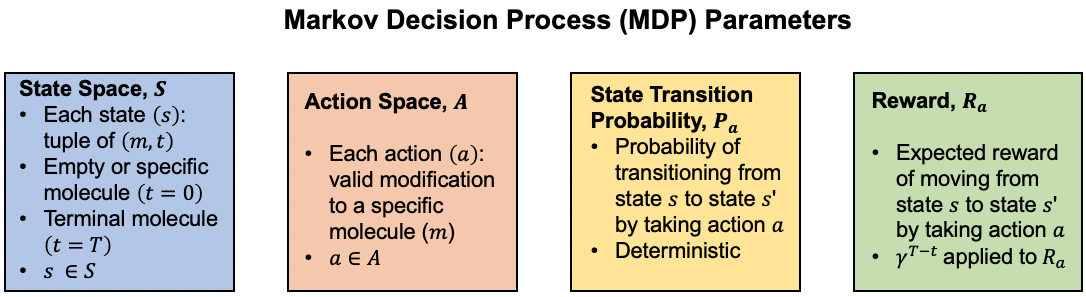


Figure 2.5: The parameters in MDP: state space (S), action space (A), state transition probability (P_a), and reward (R_a). γ, T, and t are discount factor, maximum number of steps in an episode, and number of steps taken in an episode, respectively. For further details, see Section 2.2.1.

The architecture of DQN in MolDQN consists of four-layer fully-connected network with hidden layer sizes of [1024, 512, 128, 32] and ReLU activations are used as described in Section 2.2.3. The input molecule is converted to a set of input features consisting of the bit vector from its Morgan fingerprint (Rogers and Hahn, 2010), built with a radius 3 and length 2,048 bits, and the number of steps remaining in the episode was concatenated to the vector. These layers are fully connected. The process of generating a Morgan fingerprint starts with giving an initial atom

identifier to each atom (Morgan, 1965). Then for each atom, the chemical features (*e.g.*, element and bond types) of its neighborhood, up to a given bond radius in the molecular graph (*i.e.*, in the case of MolDQN, bond radius is 3) and recorded into a bit string and compressed to a predefined length (*i.e.*, 2,048 bits for MolDQN). Each atom's identifier is then updated by the properties of the atom's environment. This process is iterated until the identifiers do not change, then duplicated identifiers are removed. The Morgan fingerprint is the remaining unique set of atom identifiers. The output dimension is the number H, which is determined by building H independent Q-functions $\{Q^{(i)} \mid i = 1, \dots, H\}$ (Section 2.2.4), where each of them are trained on different compounds generated during an episode. At each episode, a randomized value function (I. Osband *et al.*, 2019) is uniformly chosen between 1 and H, and $Q^{(i)}$ is used for decision making. This approach is combined with an ε-greedy policy, which selects the best predicted action with a probability of 1 - ε, where ε is the probability of the random action that is uniformly distributed (Tokic and Palm, 2011). Therefore, as the value of ε is annealed from 1 to 0.01 over the course of training, the probability of the predicted best action increases. ε-greedy policy and randomized value functions are used as an exploration policy. For computational efficiency, these H different models were implemented as multiple outputs on top of shared network layers. The model is trained over 5,000 episodes with the Adam optimizer (D. P. Kingma and Ba, 2014).

Adam optimizer is an adaptive optimization algorithm, which was created specifically for deep neural network training. This is a momentum-based stochastic gradient descent that estimates momentum by using exponential moving averages computed on the gradients evaluated on the current mini-batch as shown in Equations 2.35 and 2.36.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.35}$$

$$v_t = \beta_2 m_{t-1} + (1 - \beta_2) g_t^2 \tag{2.36}$$

where g_t is the gradient of the loss function f at step t with respect to w (Equation 2.39), m_t is the exponential moving average (EMA) of the first moment of the gradient g_t, v_t is the EMA of the second moment of the gradient, g_t. The loss function in MolDQN is defined in Equation 2.20. Since Adam is an iterative process, an index t is incremented by 1 at each step. Both β_1 and β_2 are exponential decay rate for the EMA and equal to 0.9 in MolDQN. Since the expectation values of the moments and the gradient value should be equal, we take the mean value of the moments as shown in Equations 2.37 and 2.38:

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{2.37}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{2.38}$$

By using all this information, Adam optimizer updates the weight vector, w, during training using the formula shown in Equation 2.39:

$$w_t = w_{t-1} - \eta \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} - \varepsilon} \tag{2.39}$$

where η is the learning rate and ε is an infinitely small value to avoid division by zero. In MolDQN, the learning rate of 0.0001 and ε of 10^{-6} are used. During each training step, the Adam optimizer calculates the gradient of the loss function (Equation 2.20) with respect to each parameter in the model. Then, it updates the EMA (Equations 2.37-2.38) and calculates the adaptive learning rate for each parameter. The weight update vector (Equation 2.39) is then calculated, and the resulting update vector is used to update the model's parameters and move them closer to the optimal set of values that minimize the loss function. Therefore, the Adam optimizer can help the model to converge faster and more accurately towards the optimal set of parameters that minimize the loss function.

During training in RL, there is a trade-off between "exploitation" and "exploration" policies due to uncertainty (Zhou *et al.*, 2019). Since there is no complete knowledge of the rewards for all the states, if the agent constantly chooses the best action that is known to produce the highest reward known as "exploitation", then the agent will never learn anything about the rewards of the other states. However, if the agent always chooses an action at random, also known as "exploration", then the agent would not receive as much reward as the agent could have achieved by choosing the best action. One of the simplest and most widely used approaches to balance these competing goals is ε-greedy policy. Although the downside of relying solely on ε-greedy policy is that this policy often wastes exploratory effort on the states that are known to be inferior. Therefore, in MolDQN, the ε-greedy policy is combined with randomized value functions to achieve deep exploration.

Zhou *et al*. used MolDQN to perform single objective molecular property optimization of either penalized logP (Section 2.2.10) or the QED (Section 2.2.11) of a designed ligand (Bickerton *et al.*, 2012). The values of penalized logP and QED range between -∞ and +∞, and 0 to 1, respectively. For both rewards, obtaining a greater positive value is more favorable.

Multi-objective optimization is performed by setting the reward for a given molecule to the length of the two-dimensional vector, $\vec{r}$, as shown in Equation 2.40:

$$\vec{r} = [QED(m), SIM(m, m_0)] \tag{2.40}$$

where m_0 is the original molecule (*i.e.*, starting molecule), m is the generated molecule, and SIM is the Tanimoto similarity computed between m_0 and m using Morgan fingerprints with radius 3 and length 2,048 bits. The similarity, SIM, is multiplied by a user-specified weight, ω, while (1 -

ω) corresponds to the weight of QED in the multi-objective optimization. This reward, $R(s)$, of state s, is calculated using Equation 2.41:

$$R(s) = \omega \times SIM(s) + (1 - \omega) \times QED(s) \tag{2.41}$$

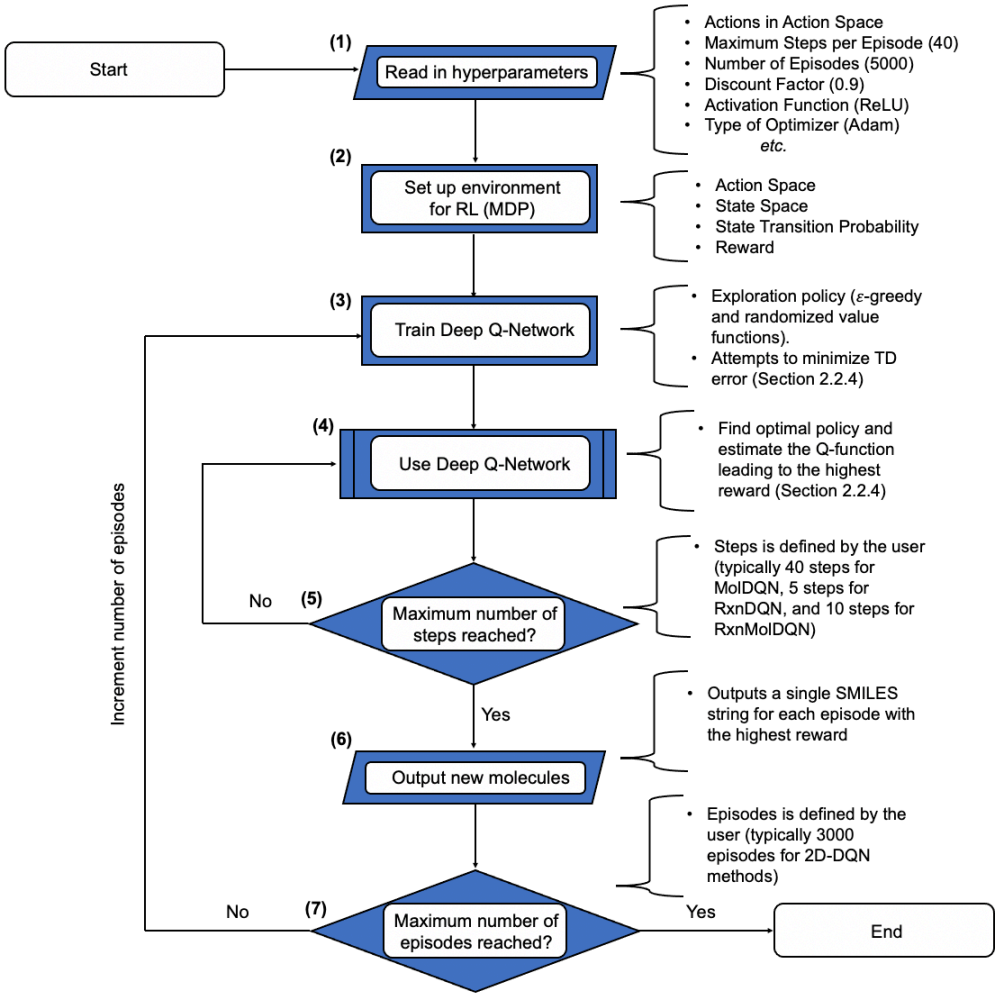


Figure 2.6: Flowchart for MolDQN, RxnDQN, and RxnMolDQN. "Start" and "End" represent the start and termination of the algorithm.

Figure 2.6 shows how MolDQN, RxnDQN, and RxnMolDQN work. The algorithm first reads in the hyperparameters from a file in JSON format **(1)**. These hyperparameters include the maximum steps per episode (the default is 40 steps), the number of episodes (the default is 5,000 steps), and

the discount factor (0.9). Then, the algorithm sets up the environment for RL **(2)** (Section 2.2.1) and trains the Deep Q-Network **(3)** (Section 2.2.3) by attempting to minimize the TD error (Section 2.2.4). TD error is the difference between the agent's predicted value and target value (Morales, 2020). The target value is obtained from the target network, which is identical in architecture to Q-Network, but only periodically updated. Once the Deep Q-Network is used to estimate the Q-function and select the most optimal action **(4)**, it checks whether the maximum number of steps (*i.e.*, maximum steps per episode) has been reached **(5)**. If yes, then it outputs a new molecule in the form of a SMILES string **(6)** and then checks whether the maximum number of episodes has been reached **(7)** (*i.e.*, number of episodes). If not, then the algorithm continues to run the Deep Q-Network until the maximum number of steps is reached. When the algorithm reaches the maximum number of episodes **(7)**, the algorithm terminates. If the maximum number of episodes has not been reached, then it increments the number of episodes and repeats by re-training the Deep Q-Network **(3)**.

MolDQN can either use naïve or bootstrap DQN (see Sections 2.3.3-2.3.4 for DQN details). Naive DQN is the conventional DQN described thus far. Bootstrap DQN differs from naïve DQN by using an ensemble of networks to compute multiple Q-values via the bootstrap (I. Osband *et al.*, 2016) to better estimate the Q-function. At the start of each episode, bootstrap DQN samples a single Q-value function from its approximate posterior. The agent then follows the policy which is optimal for that sample for the duration of the episode. Therefore, bootstrap DQN aims to encourage further exploration as compared to naïve DQN, but improvements can be limited and it is computationally expensive when training (H. Zhang *et al.*, 2020). Sampling multiple Q-networks explains the reason why bootstrap DQN takes longer to run as compared to naïve DQN, as shown in Appendix L, Tables L1-L2.

2.3.2 Reaction Deep Q-Networks (RxnDQN)

In order to improve MolDQN, I developed a new family of DQN algorithms called RxnDQN. RxnDQN was partly inspired by the poised fragment approach (Cox *et al.*, 2016). It replaces the action space of MolDQN by instead using the top 10 most commonly used reactions in medicinal chemistry (Roughley and Jordan, 2011) in combination with available reagents.

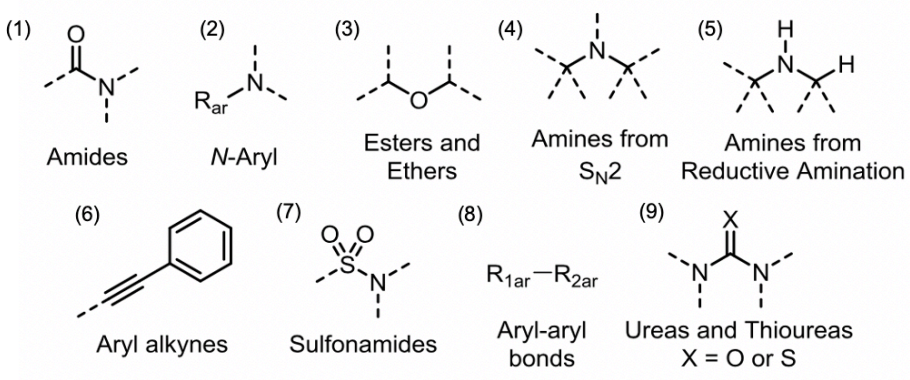


Figure 2.7: Schematics of the top 10 synthetic groups used in medicinal chemistry (Roughley and Jordan, 2011); note that the protection/deprotection groups were omitted, and ester and ether formation are counted as one group. The above synthetic groups are represented in the form of reactions using reaction SMARTS (Appendix A, Tables A1-A2).

The algorithm is designed so that a user can select any of the reactions included in Figure 2.7 (expressed in the form of synthetic groups) and can add new reactions if necessary. Since each synthetic group in Figure 2.7 is formed by a corresponding reaction (or set of reactions), and is expressed in the form of reaction SMARTS in RxnDQN, I will be referring to synthetic "reactions" not synthetic "groups" in the remaining section.

The definition of the state space, S, for RxnDQN satisfies the requirement of reversible actions in the MDP by using either forward reactions or retrosynthetic decompositions for the same reaction. MolDQN was not originally designed to perform chemical modifications to a molecule from a

particular starting chemical compound from the command line. Hence, in RxnDQN, I modified MolDQN to be able to start from a single SMILES string, m, at $t = 0$, from the command line. I implemented nine of the top 10 reactions (according to Roughley and Jordan, 2011) in my new reaction action space, with the exception of protection/deprotection reactions. Since esters and ethers group in Figure 2.7 is expressed as two types of reaction, reaction-based action space consists of 10 reactions in total. These actions were implemented using the open-source framework, RDKit (Landrum, 2019) version 2019.03.1.0 via reaction SMARTS (Appendix A, Tables A1-A2) for both the forward reaction and the corresponding retrosynthetic reaction. RxnDQN is also designed to enable users to specify the available chemical reagents used to perform these reactions. I adopted a pragmatic approach by using commercially available reagents from Reaxys (Elsevier, 2020) as described in Section 2.2.12 to obtain the reagents necessary to perform the top 10 reactions.

Each reaction in my new reaction action space implements three actions: (a) substitute R1, (b) substitute R2, and (c) one or more disconnections. The amide reaction is an exception with an additional fourth action since amide reaction is the most common reaction (Roughley and Jordan, 2011): (d) find reactive "handle" and react. All these reactions are based on the forward and disconnection reactions in Appendix A, Tables A1-A2. Figure 2.8 shows these four types of actions for amide reactions. For "substitute R1", after performing the disconnection reaction to break the amide bond, it identifies the amine synthon and performs the reactions in Appendix A, Table A2 by substituting R1 with acyl halide reagents. For "substitute R2", after performing the disconnection reaction to break the amide bond, the algorithm identifies the acid synthon and performs the reactions by substituting R2 with amine reagents. For "disconnection", the algorithm identifies the amide bond and breaks the molecule into amine and acyl halide fragments. For "find

handle and react", the algorithm searches for primary amines, carboxylates, and acyl halide scaffolds within the molecule and maps those to appropriate reagents to perform the amide formation reaction.

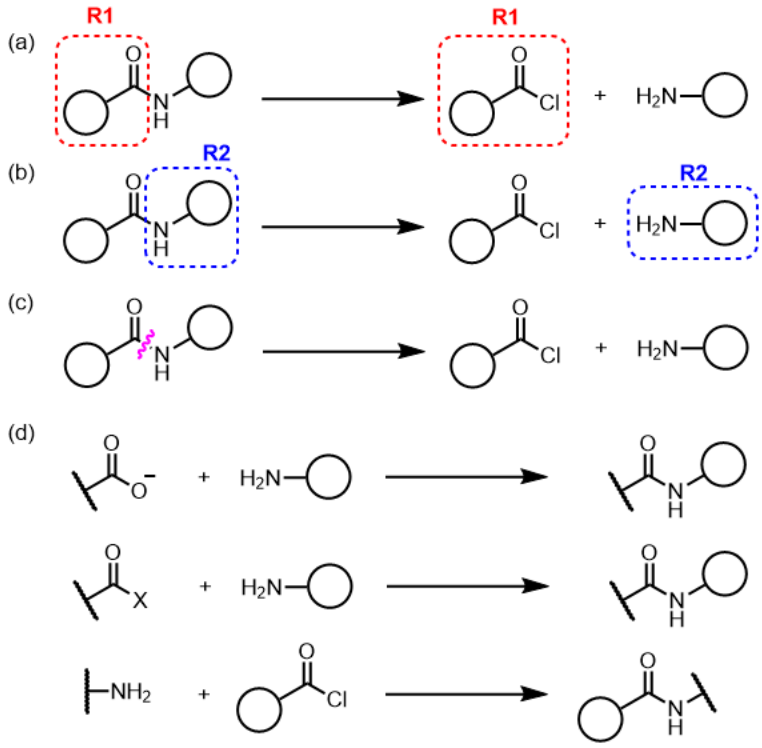


Figure 2.8: Four types of amide-related actions in the new reaction action space: (a) substitute R1, (b) substitute R2, (c) disconnection, and (d) find handle and react. The selection of these four types of amide actions occurs at random in RxnDQN.

In RxnDQN, the definition of the state transition probability, P_a, still applies as in MolDQN, but now I am making modifications to a molecule by performing chemical reactions using available reagents. The definition of the reward function remains identical to that of the MolDQN, either penalized logP or drug-likeness, QED. The user can choose to select one or more of the ten reactions from the top 10 reactions inside the reaction-based action space. When multiple reactions are selected for the reaction-based action space, the decision as to which actions are performed at each step within an episode is determined by double Q-learning (Section 2.2.4) and

would depend on the reward function the algorithm is attempting to optimize. There is also an option to select randomly the reagent from the reagent sets for each reaction using Python's `random` module to speed up the iterative process to generate new molecules. If not random, all reagents from the reagent sets will be used.

RxnDQN has a very similar flowchart to MolDQN (Figure 2.6) with the only difference being the action and state spaces. The output from RxnDQN is the number of episodes, the total time it took to perform the optimization for each episode, and the designed molecule, as a SMILES string, along with its reward value (*i.e.,* penalized logP or QED) for that particular episode, similar to MolDQN. Multi-objective optimization in RxnDQN is performed using Equation 2.41 (Section 2.3.1).

2.3.3 Hybridized Action Spaces in Deep Q-Networks (RxnMolDQN)

As mentioned in Section 2.2.14, MolDQN effectively performs a "local" exploration of chemical space and RxnDQN performs a more "global" search of chemical space. Therefore, in order to further increase the algorithm's ability to search chemical space, I explored the idea of combining the action spaces used in MolDQN and RxnDQN by developing a new hybrid *de novo* method, RxnMolDQN. This idea was inspired by the Lamarckian Genetic Algorithm (LGA) used in AutoDock4, where it was found that the hybrid global-local search method (LGA) was even more efficient than just the purely global Genetic Algorithm (GA) or local search methods alone (G. M. Morris *et al.*, 2009).

In RxnMolDQN's action space, at each step within an episode, the algorithm can choose either to perform one of the reactions specified in RxnDQN's action space or to perform the addition of

an atom or the addition or removal of a bond, as specified in MolDQN's original action space. The user defines at the start of the run which actions from each action space can be performed. The automated decision-making process for which actions to perform at each step in an episode is identical to MolDQN and RxnDQN and is being governed by the latest Q-function neural network.

In RxnMolDQN, the definition of the state transition probability, P_a, and the reward function of QED and penalized logP remain identical to that of MolDQN and RxnDQN. RxnMolDQN also follows the same flowchart as Figure 2.6. The output from RxnMolDQN is the number of episodes, the total time it took to perform the optimization for each episode, and most importantly the designed molecule, as a SMILES string, along with its reward value (*i.e.,* penalized logP or QED) for that episode, as in MolDQN and RxnDQN. Multi-objective optimization in RxnMolDQN is also performed using Equation 2.41 (Section 2.3.1).

2.3.4 Modifications to MolDQN

As described in Sections 2.3.1-2.3.3, the algorithmic differences between MolDQN, RxnDQN, and RxnMolDQN are their action and state spaces in the MDP. MolDQN can perform chemical modification without predefined sets of reagents, but RxnDQN and RxnMolDQN require user-specified sets of reagents to run the algorithm.

Additional functionalities including random sampling and a Tabu search (Glover, 1989) were added to improve the flexibility of chemical modifications that can be performed using these methods. Random sampling was implemented using NumPy (Harris *et al.*, 2020) for each reaction in the reaction-based action space. This feature was added to strike a balance between potential

chemical space exploration and computation time. The Tabu search approach was implemented using a Python dictionary by mapping the corresponding key of International Chemical Identifier (InChI) keys (Goodman *et al.*, 2021) with the value defined by the reward function. InChI keys are a fixed-length format (27 characters) directly derived from InChI where its fixed-length nature makes it suitable for data storage. InChI provides a canonical (*i.e.*, unique) identifier for chemical structures, but there is no standard to generate a canonical SMILES string (O'Boyle, 2012). Therefore, InChI keys were used in 2D-DQN methods instead of SMILES or InChI. This approach was introduced to increase computational efficiency by preventing the algorithm from re-calculating the reward of the same molecule. Results shown in Appendix L, Tables L1-L2 show that this approach is often more computationally efficient than recalculating the reward for each step in an episode.

3 Exploration of Docking Methods in Deep Q-Networks

3.1 Overview

In this chapter, I will describe the development of novel structure-based *de novo* methods that combine Deep Q-Networks with the novel action spaces I introduced in Chapter 2 and 3D structure-based methods. One major disadvantage of 2D *de novo* design methods (Sections 2.3.1-2.3.3) like MolDQN is that they do not consider the 3D interactions of the generated molecules with their target. Therefore, I incorporated a 3D structure-based approach into the Deep Q-Networks framework, starting with non-constrained docking using the open-source software, AutoDock Vina (Trott and Olson, 2010). I refer to this as "non-constrained three-dimensional Deep Q-Networks" or "nc3D-DQN". The term "non-constrained" implies that the ligands is allowed to freely translate, rotate, and change conformation (if flexible) within the active site of the protein (specified using grid box in AutoDock Vina docking) without freezing bonds or constraining a substructure of the ligand. These nc3D-DQN methods include nc3D-MolDQN, nc3D-RxnDQN, and nc3D-RxnMolDQN (Figure 1.17).

In the nc3D-DQN methods, two types of single-objective reward function are examined. These include predicted ligand efficiency (LE, Section 3.2.5) and normalized SuCOS (Sections 1.10 and 3.2.7). These methods are tested using the Malhotra dataset (Section 3.2.6).

3.2 Background

3.2.1 Introduction to Docking

Docking, also known as "protein-ligand docking" searches for optimal interactions of a small molecule in likely binding pockets of a target macromolecule and uses these to estimate its

binding affinity. In explicit structure-based virtual screening, protein-ligand docking is often used to predict the binding mode of each compound within the active site, and a scoring function is used to rank the compounds (Gilson and Zhou, 2007). While it is possible to use free energy perturbation methods to compute the binding affinity (Aldeghi *et al.*, 2016), these more rigorous methods do not always work and they are more computationally expensive than docking. Hence, more efficient, and approximate scoring functions are often used instead. In structure-based virtual screening, scoring functions are used to perform four tasks: (i) producing binding scores designed to correlate with experimental binding data ("scoring power"), (ii) correctly identify the binding mode of a ligand ("docking power"), (iii) classify molecules as either actives or inactives ("screening power"), and (iv) ranking ligands based on their binding affinity for a given target protein ("ranking power") (Su *et al.*, 2019).

Table 3.1: Comparison of features for AutoDock family (AutoDock and AutoDock Vina) and GOLD.

Programs	Algorithm	Availability	Scoring Function
AutoDock	Lamarckian GA-based	Open-source	Physics-based
AutoDock Vina	Iterated local search global optimizer	Open-source	Empirical
GOLD	GA-based	Commercial	Empirical

GOLD (Jones *et al.*, 1997) and the AutoDock family (G. M. Morris *et al.*, 2009; Ravindranath *et al.*, 2015; Trott and Olson, 2010) are examples of well-known protein-ligand docking packages that employ a single scoring function that is used to conduct all four tasks mentioned above (*i.e.*, all performed within and by the same program). These scoring functions are often referred to as "classical scoring functions" and can be categorized into four main classes: physics-based, empirical-based, knowledge-based, and machine learning-based (Section 1.5). Despite the wide use of protein-ligand docking methods to predict binding modes (often successfully), correctly

ranking ligands by their protein-ligand binding affinity using scoring functions remains challenging (Y. Li *et al.*, 2014).

A brief comparison between GOLD and AutoDock family (AutoDock and AutoDock Vina) is provided in Table 3.1. AutoDock and GOLD use stochastic optimization algorithms where AutoDock implements a choice of Monte Carlo simulated annealing search, a genetic algorithm, a hybrid global-local LGA (Section 2.3.3) or a Solis-Wets local search. In the case of GOLD, a genetic algorithm is used. For AutoDock Vina, an iterated local search global optimizer is used. AutoDock Vina performs flexible docking by using this stochastic search, where a series of steps are taken, and each step consists of a mutation and local optimization, and Metropolis criterion is used to either accept or reject each step. This mutation involves changes to the position, orientation, and torsions of the docked molecule. Another major difference between GOLD and the AutoDock family is their availability. The AutoDock family is free and open-source for academic use, whereas GOLD is only commercially available. In addition, AutoDock uses physics-based scoring function, whereas AutoDock Vina and GOLD use empirical scoring function. The scoring functions used in the AutoDock family are described in Section 3.2.3.

3.2.2 Machine Learning in Docking

In early applications of machine learning in a docking scenario, a protein-ligand interaction fingerprint was used as an input to decide whether the ligand is a binder or not (Ballester and Mitchell, 2010; Durrant and McCammon, 2011; Klon *et al.*, 2004). Although these methods were reported to perform better than the classical scoring functions, there was a concern due to lack of sensitivity for pose changes and lack of generalization to new targets (Gabel *et al.*, 2014; Kramer and Gedeck, 2010).

Recent advances in machine learning have led to the development of a new class of scoring functions that are often reported to outperform classical scoring functions on standard benchmarks, by improving their ability to rank compounds by their binding affinity. Imrie *et al.* (Imrie *et al.*, 2018) improved upon the CNN introduced by Ragoza *et al*. (Ragoza *et al.*, 2017) and showed that transfer learning could be used to improve target-specific models, while Boyles *et al.* (Boyles *et al.*, 2020) showed that by augmenting structure-based features with a rich set of ligand-based features, binding affinity prediction could be improved by using random forest regression. Additionally, deep learning-based scoring functions for binding affinity prediction have been shown to have significantly better correlation with experimental binding data than classical scoring functions, including AEScore (Meli *et al.*, 2021) and protein-ligand interaction graphs or PLIGs (Marc A. Moesser *et al.*, 2022). Scantlebury *et al.* identified key binding site interactions and binding pharmacophores from a protein target using graph neural network (Scantlebury *et al.*, 2022).

Several recent works have applied machine learning to perform tasks based on docking scores and predicted binding modes (Gentile *et al.*, 2020; Graff *et al.*, 2021; Jeon and Kim, 2020; Lyu *et al.*, 2019; M. Thomas *et al.*, 2021). For example, DockStream (Guo *et al.*, 2021) combines automated docking workflow in conjunction with the *de novo* design algorithm, REINVENT in Section 1.7 (Blaschke *et al.*, 2020), to incorporate docking in the design.

There is also a trend towards development of methods that enable the identification of features within a ligand that contributes to binding. Brown *et al.* predict the influence of substructures on binding affinity relative to other ligands in a congeneric ligand series used as an input (B. P.

Brown *et al.*, 2021). Hochuli *et al.* developed several methods to visualize the impact of specific atoms on the predicted binding by a CNN (Hochuli *et al.*, 2018). Recently, Sanchez-Lengeling *et al.* proposed evaluation metrics for attribution methods to facilitate the development of more interpretable deep learning-based methods (Sanchez-Lengeling *et al.*, 2020).

As a starting point to my fully automated *de novo* design and docking pipeline, I investigated the use of the docking score from AutoDock Vina (Trott and Olson, 2010) in the reward function in a new family of deep learning-based *de novo* methods called "nc3D-DQN" (Section 3.3.1). This structure-based approach enhances our ability to design novel molecules that are more likely to be complementary to a desired binding pocket by keeping the key binding site interactions, whilst rejecting non-complementary molecules, as well as harnessing unused pockets and improving the scaffold-hopping performance (Guo *et al.*, 2021).

3.2.3 AutoDock Vina

AutoDock Vina used in the nc3D-DQN methods is one of the most widely used open-source docking programs (Trott and Olson, 2010) and is currently maintained and developed by the Forli Lab (The Scripps Research Institute). Users must prepare the structures of the molecules being docked (both the receptor and the ligand) and specify the search space (*i.e.*, grid box dimensions for x, y, and z in AutoDock Vina).

AutoDock Vina was developed based on the AutoDock 4 program (G. M. Morris *et al.*, 2009), but uses a different scoring function and optimization algorithm (Section 3.2.1: Note that AutoDock 4 is a version of AutoDock). The scoring function used in AutoDock Vina is shown in Equation 3.1, and uses an empirical scoring function to predict the protein-ligand binding free

energy, s_1, from the intermolecular part (c_{inter1}) of the lowest-scoring conformation (designated as 1).

$$s_1 = g(c_1 - c_{intra1}) = g(c_{inter1}) \tag{3.1}$$

$$c = c_{inter} + c_{intra} \tag{3.2}$$

where c is the sum of intermolecular (c_{inter}) and intramolecular (c_{intra}) contributions (Equation 3.2) and g is an arbitrary strictly increasing non-linear function. In the AutoDock Vina output, other low-scoring conformations (i) are also formally given s values, but to preserve the ranking, these conformations use c_{intra} of the best binding mode found in a given search as shown in Equation 3.3:

$$s_i = g(c_i - c_{intra1}) \tag{3.3}$$

The conformation-dependent part of the scoring function in AutoDock Vina is designed to work with Equation 3.4:

$$c = \sum_{i<j} f_{t_i t_j}(r_{ij}) \tag{3.4}$$

where the summation is over all the pairs of atoms i and j that can move relative to one another, normally excluding atoms separated by three consecutive covalent bonds. In this equation, each atom i is assigned an atom type t_i and a symmetric set of interaction functions $f_{t_i t_j}$ of the interatomic distance r_{ij} need to be defined. This interaction functions of $f_{t_i t_j}$ are defined relative to the surface distance (Equation 3.5) as shown in Equation 3.6.

$$d_{ij} = r_{ij} - R_{t_i} - R_{t_j} \tag{3.5}$$

$$f_{t_i t_j}(r_{ij}) \equiv h_{t_i t_j}(d_{ij}) \tag{3.6}$$

where R_t is the van der Waals radius of atom type t. In this scoring function, $h_{t_i t_j}$ is a weighted sum of steric interactions ($gauss_1$, $gauss_2$, and $repulsion$), which is identical for all atom pairs,

hydrophobic interaction between hydrophobic atoms and hydrogen bonding (where applicable). These steric terms are defined in Equations 3.7-3.9:

$$gauss_1(d) = e^{-(\frac{d}{0.5\text{Å}})^2} \tag{3.7}$$

$$gauss_2(d) = e^{-(\frac{(d-3\text{Å})}{2\text{Å}})^2} \tag{3.8}$$

$$repulsion(d) = \begin{cases} d^2 & if\ d < 0 \\ 0 & if\ d \geq 0 \end{cases} \tag{3.9}$$

where d is defined in Equation 3.5. The hydrophobic term equals 1 when $d < 0.5$ Å and equals 0 when $d > 1.5$ Å (linearly interpolated between these values). The hydrogen bonding term equals 1 when $d < -0.7$ Å and equals 0 when $d > 0$ Å (linearly interpolated between these values). In this implementation, all interaction functions of $f_{t_i t_j}$ have a cut-off at $r_{ij} = 8$ Å. Lastly, the conformation-independent function g in Equation 3.1 is defined as:

$$g(c_{inter1}) = \frac{c_{inter}}{1 + \omega N_{rot}} \tag{3.10}$$

where N_{rot} is the number of active rotatable bonds between heavy atoms in the ligand and ω is the associated weight. Details of the associated weight can be found in the original work (Trott and Olson, 2010).

In the case of AutoDock 4, the scoring function used to predict protein ligand binding free energy is based on a semi-empirical free energy force field (G.M. Morris *et al.*, 2014). This force field was parameterized using a large number of protein-inhibitor complexes where both structure and inhibition constants (K_i) are known. The binding is evaluated in two steps. The ligand and protein first start in an unbound conformation. Then in the first step, the intermolecular energetics are estimated due to transition from these unbound states to the conformations of the protein and ligand in the bound state. This first step is described in Equation 3.11 where the force field

includes six pair-wise evaluations, V, and an estimate of the conformational entropy lost upon binding, ΔS_{conf}.

$$\Delta G = \left(V_{bound}^{L-L} - V_{unbound}^{L-L}\right) + \left(V_{bound}^{P-P} - V_{unbound}^{P-P}\right) + \left(V_{bound}^{P-L} - V_{unbound}^{P-L} + \Delta S_{conf}\right) \quad (3.11)$$

where P is the protein and L is the ligand. The second step evaluates the intermolecular energetics (dispersion/repulsion, hydrogen bonding, electrostatics, and desolvation) of combining the ligand and protein in their bound conformation as shown in Equation 3.12.

$$V = W_{vdw} \sum_{i,j} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right) + W_{hbond} \sum_{i,j} E(t) \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) + W_{elec} \sum_{i,j} \frac{q_i q_j}{e(r_{ij}) r_{ij}} + W_{elec} \sum_{i,j} \frac{q_i q_j}{e(r_{ij}) r_{ij}} + W_{solv} \sum_{i,j} \left(S_i V_j + S_i V_i\right) e^{\left(\frac{r_{ij}^2}{2\sigma^2}\right)} \quad (3.12)$$

where W are coefficient that have been obtained by a linear regression to predict the empirical free energy based on a set of experimentally determined binding constants. The first term is for van der Waals interaction (dispersion/repulsion) and is based on the Amber force field. The second term is for directional H-bonding. Parameters C and D provide a maximal well depth of 5 kcal/mol at 1.9 Å for hydrogen bonds with oxygen and nitrogen, and a well depth of 1 kcal/mol at 2.5 Å for hydrogen bonds with sulphur. $E(t)$ provides directionality based on the angle t from ideal H-bonding geometry. The third term is a screened Coulomb potential for electrostatics. The final term is a desolvation potential based on the volume of atoms, V, that surround a given atom and shelter it from solvent. This is weighted by a solvation parameter, S, and an exponential term with distance-weighting factor of σ = 3.5 Å. Detailed presentation of these functions can be found in the AutoDock 4 manual (G.M. Morris *et al.*, 2014).

There are several advantages of using AutoDock Vina over its predecessor AutoDock 4. AutoDock Vina does not require the calculation of external grip map files (which are built-in and automatically calculated), nor does it require the assignment of partial charges. In AutoDock 4,

grid maps are required for each atom type present in the ligand being docked. These maps are calculated using AutoGrid (part of the AutoDock suite). A grid map consists of 3D lattice of regularly space points, which is surrounding (either entirely or partly) and centered on some region of interest in a macromolecule. Typically, grid point spacing varies from 0.2 to 1.0 Å, but the default is 0.375 Å. Each point within the grid map stores the potential energy of a 'probe' atom or functional group, which is due to all the atoms in the macromolecule. Partial charges that are specified by the user in AutoDock 4 are ignored for AutoDock Vina. This is because AutoDock Vina has its own way of dealing with electrostatic interactions through the hydrophobic and hydrogen bonding terms. Moreover, since AutoDock Vina uses multiple CPUs or CPU cores, it shortens the running time by sometimes orders of magnitude as compared to AutoDock 4. Typically, for my nc3D-DQN methods, AutoDock Vina runs take a few seconds to complete on current hardware (CPU: 48 core Cascade Lake (Intel Xeon Platinum 8,268 CPU @ 2.90 GHz); Memory: 392 GB (Richards, 2015)). Finally, it is also reported that AutoDock Vina improves the average accuracy for binding mode predictions as compared to AutoDock 4 (Trott and Olson, 2010).

3.2.4 Open Babel

Open Babel is an open-source chemical toolbox that can be used in C++, Python, Java, Ruby, Perl, and C#. There are many applications of Open Babel including conversion of multiple chemical file formats, conformer searching, 2D depiction, filtering, substructure and similarity searching. This toolbox is widely used to handle chemical data in areas such as molecular modelling and cheminformatics (O'Boyle *et al.*, 2011). Further details regarding Open Babel can be found in the documentation (O'Boyle *et al.*, 2020).

In the nc3D-DQN methods, Open Babel is used to convert the SMILES strings of the latest *de novo*-designed molecule into a 3D molecule in PDBQT format, which is used to prepare the ligand for AutoDock Vina (Trott and Olson, 2010).

3.2.5 Predicted Ligand Efficiency (LE)

Predicted ligand efficiency (LE) is one of the single objective optimization reward functions used in the nc3D-DQN methods. LE is widely used in drug discovery (Hopkins *et al.*, 2004) and here, it is calculated by normalizing the predicted binding affinity from AutoDock Vina (in kcal/mol) by the number of heavy atoms as computed by RDKit; the units of LE used in the nc3D-DQN methods are kcal $mol^{-1}HA^{-1}$ (Equation 3.13). Heavy atoms (HA) are those atoms whose atomic number is greater than 1.

$$Ligand\ efficiency\ (\triangle\ g) = \frac{\triangle\ G}{N_{non-hydrogen\ atoms}}\ (kcal\ mol^{-1}HA^{-1}) \qquad (3.13)$$

LE is often used to identify fragments and lead compounds with optimal combinations of pharmacological and physicochemical properties (Abad-Zapatero and Metz, 2005). Typically, LE of 0.3 kcal $mol^{-1}HA^{-1}$ are considered satisfactory to be optimized for a drug that satisfy Lipinski rules for oral bioavailability (Schultes *et al.*, 2010). Note that the sign of LE in my implementation is reversed, since the DQN framework uses a hill climbing algorithm.

3.2.6 Malhotra Dataset

The Malhotra set is a collection of 297 ligand-pairs solved in complex with the same protein, where pairs (smaller and larger ligands) differ in size but have a substructure in common

(Malhotra and Karanicolas, 2017). I chose to use the Malhotra set to test the nc3D-DQN methods for their ability to discover the larger ligand when starting from the smaller ligand of the same ligand-pair (Figure 3.1). If this is possible, then it implies that the nc3D-DQN methods are useful tools in fragment-based drug discovery.

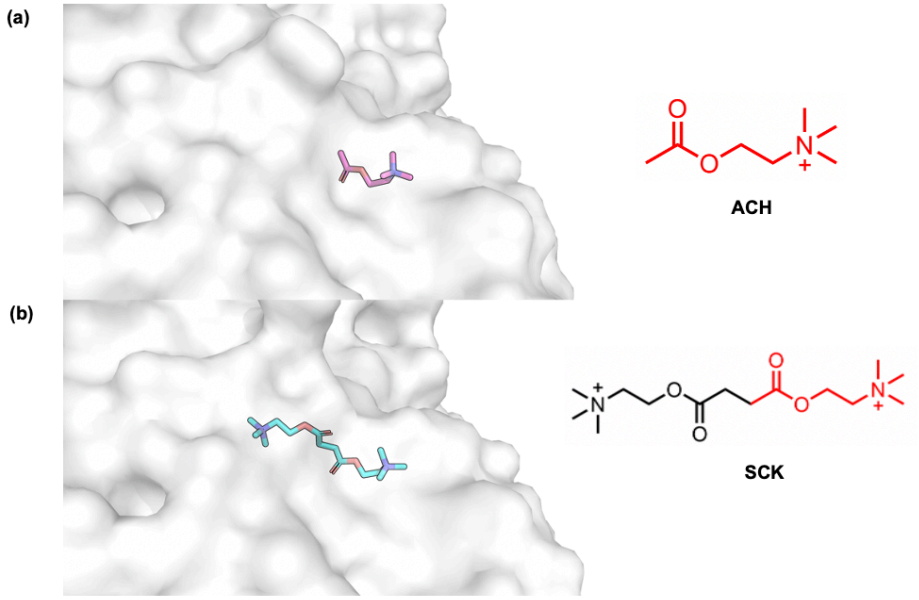


Figure 3.1: An example of smaller **(a)** and larger **(b)** ligand-pair from the Malhotra set using 2ha4 and 2ha2, respectively. The bound ligands are ACH (magenta) and SCK (cyan). These figures are created using PyMOL (Schrödinger, 2022). The common substructures between these ligands are highlighted in red. This figure shows that the smaller ligand is growing to the larger ligand.

In order to test the performance of nc3D-DQN methods (Section 3.3.1) using LE or normalized SuCOS as the reward function, 51 and 45 ligand-pairs from the Malhotra set, respectively, were selected using the procedures described in Section 5.2. The smaller ligand of these ligand-pairs was used as the starting molecule in each of the various nc3D-DQN methods.

3.2.7 Normalized SuCOS

Normalized SuCOS (Leung and Moesser, 2021) is another structure-based single objective optimization reward function used in the nc3D-DQN methods. SuCOS is a Combined Overlap

Score. It is an evenly weighted combination of shape overlap and pharmacophoric-feature overlap between two molecules, and it is normalized such that 1 corresponds to perfect shape and pharmacophoric overlap (Figure 3.2(a)) while 0 means no overlap at all (Figure 3.2(b)).

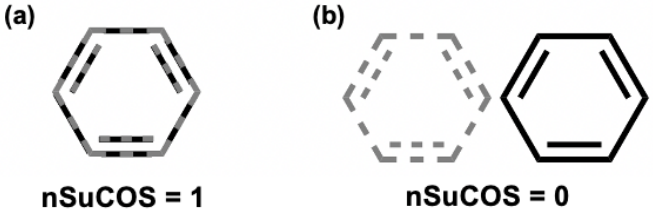


Figure 3.2: Schematics of normalized SuCOS score: **(a)** score of 1 and **(b)** score of 0. The solid and dashed structures represent two separate molecules.

This reward is considered to be a pharmacophoric-based reward function since it incorporates shape overlap and pharmacophoric overlap as discussed in Section 1.10 (Leung *et al.*, 2019). According to International Union of Pure & Applied Chemistry (IUPAC), a pharmacophore is defined as "the ensemble of steric and electronic features that is necessary to ensure the optimal supra-molecular interactions with a specific biological target structure and to trigger (or to block) its biological response" (Wermuth *et al.*, 1998). Pharmacophoric features include hydrogen-bond acceptors (HBA) and donors (HBD), positive and negative charges, positive and negative ionizable groups, aromatic planes, and aliphatic and aromatic hydrophobic groups (Gritsch *et al.*, 2001). These features are important since they describe how the small molecules are believed or known to interact with the target (*i.e.*, making it 'active') and this requires developing a pharmacophoric hypothesis, which can be useful in discovering new drugs or scaffold hopping (Kaserer *et al.*, 2015). Therefore, the nc3D-DQN methods with normalized SuCOS is an alternative to the nc3D-DQN methods with LE, since it introduces a pharmacophoric component.

In the normalized SuCOS reward function (Section 1.10), the reference molecule is the larger ligand (*i.e.*, known active compound) from the Malhotra set (Malhotra and Karanicolas, 2017) and the query molecule is the generated molecules from the nc3D-DQN methods. The larger ligand is effectively the desired ligand that we would like the *de novo* design method to mimic. SuCOS can be used like Tanimoto similarity, but here we are computing similarity to the desired actives in 3D. The advantage of using this type of reward function over LE (Section 3.2.5), *e.g.*, is that it enables the incorporation of pharmacophoric hypotheses in *de novo* design, thus potentially facilitating bioisosteric discovery.

Bioisosteres are chemical substituents or groups with common physicochemical properties that elicit similar biological activity (Patani and LaVoie, 1996). In drug discovery, the purpose of bioisosteric discovery is to affect intrinsic potency and selectivity, to influence conformation, to solve problems associated with drug development like modulating basicity, solubility, lipophilicity, and P-glycoprotein recognition; and to address issues associated with toxicity and metabolism (Meanwell, 2015). Pharmacophoric reward functions can aid in bioisosteric discovery because bioisosteres are structurally different molecules or substructures that can form comparable intermolecular interactions (Wood *et al.*, 2012). Therefore, fragments that bind to similar protein structures could exhibit a degree of bioisosterism in the case of normalized SuCOS.

In terms of performance of SuCOS, SuCOS tends to produce fewer false positives and false negatives than Root Mean Square Deviation (RMSD) and Protein Ligand Interaction Fingerprint (PLIF) similarity in redocking (Leung *et al.*, 2019). In addition, when cross-docking, SuCOS differentiates the experimentally observed binding modes of an elaborated molecule better than RMSD and PLIF similarity. Furthermore, in virtual screening, SuCOS has been show to

outperform the AutoDock Vina scoring function at differentiating actives from decoys in the DUD-E dataset.

3.2.8 Project Aim

In Chapter 2, I stressed the idea that the ability to explore chemical space efficiently is essential to drug discovery. This led to the development of reaction-based Deep Q-Networks, RxnDQN and RxnMolDQN. However, these 2D-DQN methods do not consider the interaction of the newly generated molecules with the target protein's active site, which is another important aspect in drug discovery.

Table 3.2: Summary of *de novo* methods developed for the nc3D-DQN methods, and which action spaces and reactions are used, together with their reward functions. All methods noted here were developed by the author. LE, nSuCOS, and T stand for predicted ligand efficiency, normalized SuCOS, and Tanimoto similarity, respectively. LE is calculated from AutoDock Vina's predicted binding affinity (Trott and Olson, 2010). nSuCOS is computed using the method described in Section 3.2.7 (Leung and Moesser, 2021).

Method	Atom/Bond Action Space	Reactions Action Space	Docking	Single Objective Reward Function	Multi-Objective Reward Function
nc3D-MolDQN	Yes	No	Yes (Non-constrained)	LE	LE + T
nc3D-MolDQN	Yes	No		nSuCOS	nSuCOS + T
nc3D-RxnDQN	No	Yes		LE	LE + T
nc3D-RxnDQN	No	Yes		nSuCOS	nSuCOS + T
nc3D-RxnMolDQN	Yes	Yes		LE	LE + T
nc3D-RxnMolDQN	Yes	Yes		nSuCOS	nSuCOS + T

Therefore, I developed the nc3D-DQN methods by incorporating AutoDock Vina (Trott and Olson, 2010) with two new types of reward: predicted ligand efficiency, LE, and normalized SuCOS. Since the nc3D-DQN methods use the same action spaces as their 2D counterparts, they allow for compound generation in "local" and/or "global" exploration of chemical space, depending on the method (Section 2.2.14), whilst simultaneously proposing molecules that fit in the target protein's binding pocket.

Here, I investigate how the incorporation of a docking component (Section 3.2.5) and pharmacophoric-based features (Section 3.2.7) into Deep Q-Network-based *de novo* algorithms can guide molecular design by comparing the atom/bond-based action space of nc3D-MolDQN, reaction-based action space of nc3D-RxnDQN, and hybridized action space of nc3D-RxnMolDQN. These results are described in Chapter 6. Table 3.2 shows a summary of the nc3D-DQN methods that I developed (three new methods (nc3D-MolDQN, nc3D-RxnDQN, and nc3D-RxnMolDQN) where each was tested with four new reward functions (LE, nSuCOS, LE + T, and nSuCOS + T)).

3.3 Methodology

3.3.1 Non-Constrained Docking in Deep Q-Networks (nc3D-DQN)

As explained earlier in Section 3.2, the nc3D-DQN methods with LE as the reward function are 3D structure-based versions of 2D-DQN methods (Sections 2.3.1-2.3.3). This type of nc3D-DQN methods is the basis of nc3D-DQN methods with normalized SuCOS (nSuCOS). The nc3D-DQN methods with LE and normalized SuCOS as the reward will be referred to as “nc3D-DQN (LE)” and “nc3D-DQN (nSuCOS)” methods, respectively.

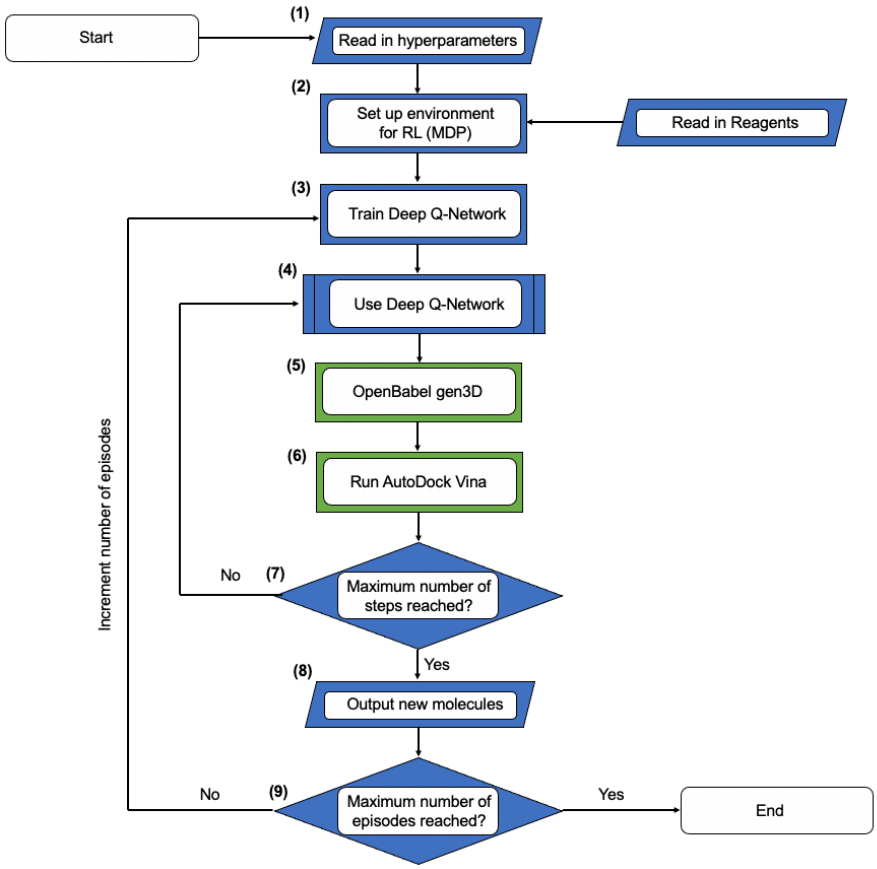


Figure 3.3: Flowchart for nc3D-DQN (LE) methods (3D-MolDQN, 3D-RxnDQN, and 3D-RxnMolDQN), where unique workflow to nc3D-DQN methods is highlighted in green. Details for the parts in blue can be found in Figure 2.6.

The nc3D-DQN (LE) methods (Figure 3.3) are similar to their 2D counterparts (Figure 2.6): they use the same action spaces, but differ at step **(5)**, where the generated SMILES string is converted into a 3D molecule (in PDBQT format) using OpenBabel's `gen3D` (O'Boyle *et al.*, 2011). The PDBQT file is then used as the input ligand in an AutoDock Vina docking in step **(6)** to the previously prepared protein PDBQT file. The algorithm then selects the docked pose with the best estimated binding affinity and uses this predicted affinity as part of the reward function (Section 3.2.5). The output PDBQT file from the last step is kept within each episode.

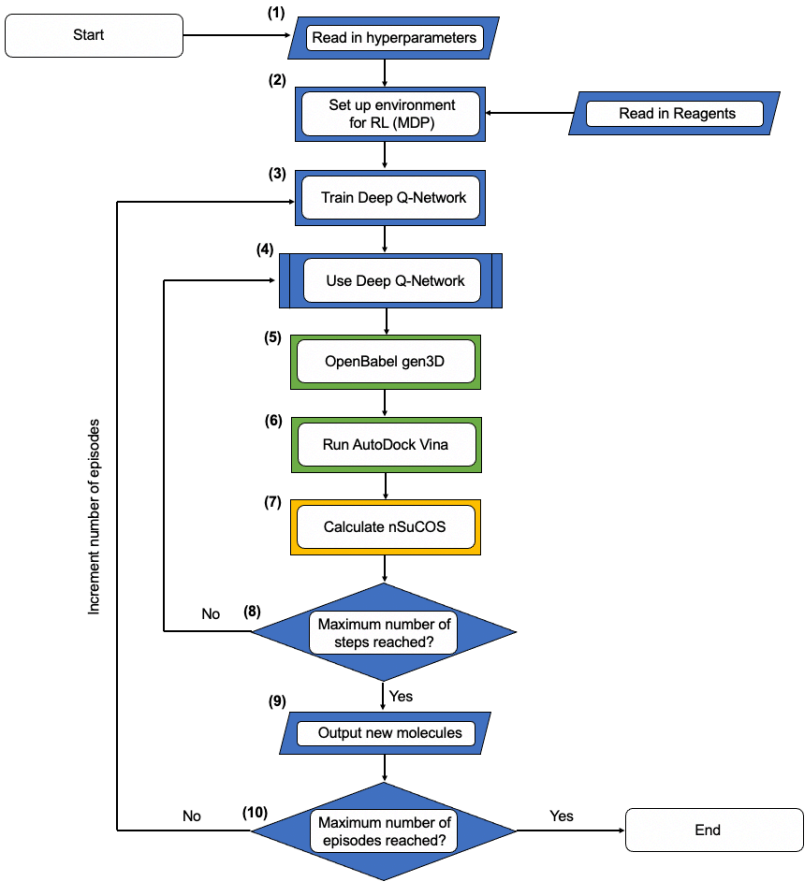


Figure 3.4: Flowchart for nc3D-DQN (nSuCOS) methods (3D-MolDQN, 3D-RxnDQN, and 3D-RxnMolDQN), where the steps unique to nc3D-DQN and nc3D-DQN (nSuCOS) methods are highlighted in green and yellow, respectively. Details for the parts in blue can be found in Figure 2.6.

In the case of nc3D-DQN (nSuCOS) methods (Figure 3.4), the first six steps are identical to Figure 3.3, but after performing step **(6)** (*i.e.*, Run AutoDock Vina), the algorithm computes for the nSuCOS score in step **(7)**. The nSuCOS score (Section 1.10) is calculated from the SDF file generated by OpenBabel from the output PDBQT file produced in step **(6)** using OpenBabel's `gen3D` (*i.e.*, the query molecule for nSuCOS), and the pre-processed larger ligand's SDF file (*i.e.*, the reference molecule for nSuCOS).

3.3.2 Comparison to 2D-DQN Methods

The key difference between 2D-DQN (Sections 2.3.1-2.3.3) and nc3D-DQN (Section 3.3.1) methods is the reward function. AutoDock Vina attempts to predict the protein-ligand binding affinity, ΔG_{bind}, which for ligands that bind is a negative quantity with units in kcal/mol. As described earlier in Section 3.2.5, the reward function for the nc3D-DQN (LE) methods is based on the molecule's predicted ligand efficiency. The benefit of using the predicted ligand efficiency over binding affinity is that it normalizes for molecular size and can help to avoid molecular obesity (Hann, 2011; Hopkins *et al.*, 2014). The term 'molecular obesity' is used when molecules are grown too large and become too lipophilic while optimizing for potency. At a given step within an episode, the algorithm computes the predicted ligand efficiency after docking, but this value is multiplied by -1 due to maximizing nature of the algorithm. If the process fails at any step, say at file conversion, or at docking, the algorithm is designed to return a predicted binding affinity (or LE) of 0.0. For the nc3D-DQN (nSuCOS) methods, the nSuCOS (Section 3.2.7) reward function is used as described in Section 3.3.1.

3.3.3 Multi-objective Optimization in nc3D-DQN Methods

The general form for the reward function of the multi-objective optimization models for the nc3D-DQN (LE) methods is shown in Equation 3.14:

$$R_m = \omega a_m + (\omega - 1) b_m \tag{3.14}$$

where m is the generated molecule at the current step within an episode, ω is a weight that ranges from 0 to 1, and R_m is the reward of molecule, m. For the nc3D-DQN (LE) methods, I used Tanimoto similarity as a_m and the predicted ligand efficiency (Section 2.3.5) as b_m ($b_m < 0$ kcal $mol^{-1}HA^{-1}$). I multiply b_m by -1 since favorable ligand efficiency is generally a negative quantity, and desirable values move in the opposite direction to Tanimoto similarity. Tanimoto similarity

was used to incorporate 2D-based metrics and to increase the methods' ability to generate molecules closer to the target molecules in the Malhotra set. Tanimoto similarity is computed in a similar way as Equation 2.41 (*i.e.*, generated molecule and Malhotra smaller ligand).

In the case of nc3D-DQN (nSuCOS) methods, the reward function for multi-objective optimization is expressed as:

$$R_m = \omega a_m + (1 - \omega) b_m \tag{3.15}$$

where a_m is Tanimoto similarity, b_m is normalized SuCOS (Section 3.2.7), and the remaining variables are identical to Equation 3.14. In the case of nc3D-DQN (nSuCOS) methods, Tanimoto similarity was computed between the generated molecule and Malhotra larger ligand (*i.e.*, target molecule).

4 Constrained Docking in Deep Q-Networks: SARS-CoV-2 Main Protease

4.1 Overview

In this chapter, I will describe the extension of the nc3D-DQN methods (Section 3.3.1) by incorporating constrained docking (Section 4.2.1) using the open-source molecular docking program, AutoDock Vina (Trott and Olson, 2010). Constrained docking is when a substructure of the ligand is constrained to a known binding mode and is closely related to covalent docking to flexible sidechain like Cys-145 in SARS-CoV-2 main protease. Therefore, c3D-DQN methods enable a more constrained 3D design compared to the nc3D-DQN methods thus enhancing confidence in the predicted binding pose of the docked ligand in the active site of a protein. This extension is referred to as constrained three-dimensional Deep Q-Networks or "c3D-DQN". These c3D-DQN methods include c3D-MolDQN, c3D-RxnDQN, and c3D-RxnMolDQN (Figure 1.17).

4.2 Background

4.2.1 Constrained Docking Versus Covalent Docking

There are two types of constrained docking: covalent ligand design and fragment elaboration (Figure 4.1). The first case is to chemically modify a side chain in the protein (Figure 4.1(a)). For example, in SARS-CoV-2 main protease, the generated molecules are constrained to bind to Cys-145 by covalent docking. Covalent docking enables a covalently attached ligand to bind to a user-defined amino acid in the active site of a target protein of interest and explores the resultant constrained degrees of freedom to find the best covalent binding modes. The same approach can be used for "constrained" docking, in particular for fragment-based *de novo* drug design where

the fragment is constrained to a known binding mode (Figure 4.1(b)). In this case, a non-covalent fragment of SARS-CoV-2 main protease X-ray crystal structure could be used as a starting point for fragment elaboration. Therefore, the case study with SARS-CoV-2 main protease (results in Chapter 6) is both constrained and covalent, and I refer to this as constrained/covalent docking.

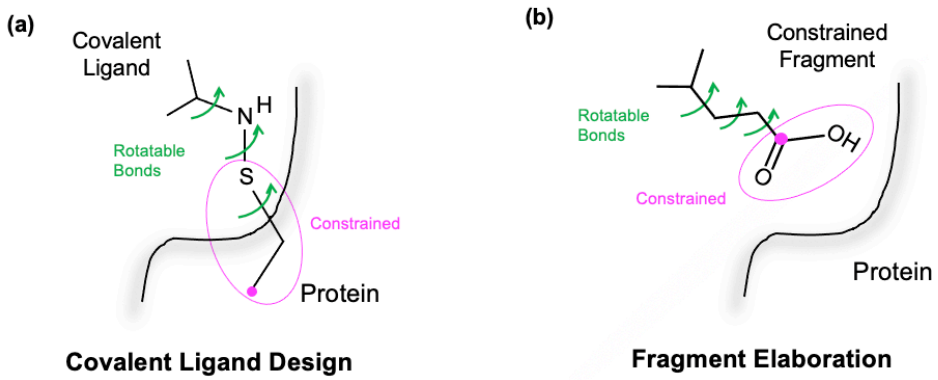


Figure 4.1: Schematic for two potential types of constrained docking: **(a)** protein/amino acid side chain is covalently modified; **(b)** constrained fragment of the molecule. In **(a)**, covalent ligand is bound covalently to cysteine side chain. In (b), the carboxylate (pink) here would have come from X-ray crystallography and would be a non-covalent fragment. SARS-CoV-2 main protease case study is shown using type (a).

4.2.2 SARS-CoV-2 Main Protease

Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is the cause of coronavirus disease 2019 (COVID-19), which the World Health Organization declared as a pandemic on March 11th, 2020. As of December 13th, 2022, there have been over 640 million confirmed cases of COVID-19, including over 6.6 million deaths (World-Health-Organization, 2022).

In the SARS-CoV-2 viral life cycle, polyproteins pp1a and pp1b are cut by two viral proteases into its constituent viral proteins, specifically SARS-CoV-2 main protease (M^{pro}) and papain-like

protease (PL^{pro}). M^{pro} and PL^{pro} cut at eleven (Table 4.1) and three cleavage sites, respectively. SARS-CoV-2 main protease is 96% identical to SARS-CoV M^{pro} (Morse *et al.*, 2020).

Table 4.1: The eleven cleavage sites for SARS-CoV-2 main protease taken directly from Chan *et al.* (H. T. H. Chan *et al.*, 2021). s01-s11 are the corresponding 11-residue peptides, blue/red demonstrates positively/negatively charged residues, histidine is shown in purple, polar side chains are shown in green, and cysteine is shown in yellow. Note the exclusive preference for Q at P1.

	P6	P5	P4	P3	P2	P1	P1′	P2′	P3′	P4′	P5′
s01	T	S	A	V	L	Q	S	G	F	R	K
s02	S	G	V	T	F	Q	S	A	V	K	R
s03	K	V	A	T	V	Q	S	K	M	S	D
s04	N	R	A	T	L	Q	A	I	A	S	E
s05	S	A	V	K	L	Q	N	N	E	L	S
s06	A	T	V	R	L	Q	A	G	N	A	T
s07	R	E	P	M	L	Q	S	A	D	A	Q
s08	P	H	T	V	L	Q	A	V	G	A	C
s09	N	V	A	T	L	Q	A	E	N	V	T
s10	T	F	T	R	L	Q	S	L	E	N	V
s11	F	Y	P	K	L	Q	S	S	Q	A	W

According to crystallographic and computational modelling studies for M^{pro} (Świderek and Moliner, 2020), it is proposed that histidine-41 (*i.e.*, His-41) deprotonates the cysteine-145 (*i.e.*, Cys-145) during M^{pro} catalysis. The resulting thiol reacts with the carbonyl of the scissile amide to form an acyl-enzyme intermediate. Then, this intermediate is stabilized by a hydrogen bonding interaction that keeps the scissile amide carbonyl in an 'oxyanion hole' (formed by residues 143-145). At this stage, the product's C-terminal part is likely to leave the protein's active site. This acyl-enzyme intermediate gets hydrolyzed with a loss of the N-terminal product, which generates the active M^{pro} (Figure 4.2(a)). Inhibitors of SARS-CoV-2 M^{pro} can be either non-covalent or covalent – indeed, the first crystal structure of SARS-CoV-2 M^{pro} was showed with the covalent inhibitor N3 (Z. Jin *et al.*, 2020), and Pfizer's clinically approved SARS-CoV-2 M^{pro} inhibitor nirmatrelvir binds covalently, both to Cys-145. Therefore, by retaining covalent interactions between Cys-145 and any newly generated molecules from *de novo* design method, we can design

inhibitors of SARS-CoV-2 main protease. Chan *et al.* (H. T. H. Chan *et al.*, 2021) showed that subsites S4, S2, S1, and S1′ were most important in SARS-CoV-2 M^{pro}'s active site (Figure 4.2(b)).

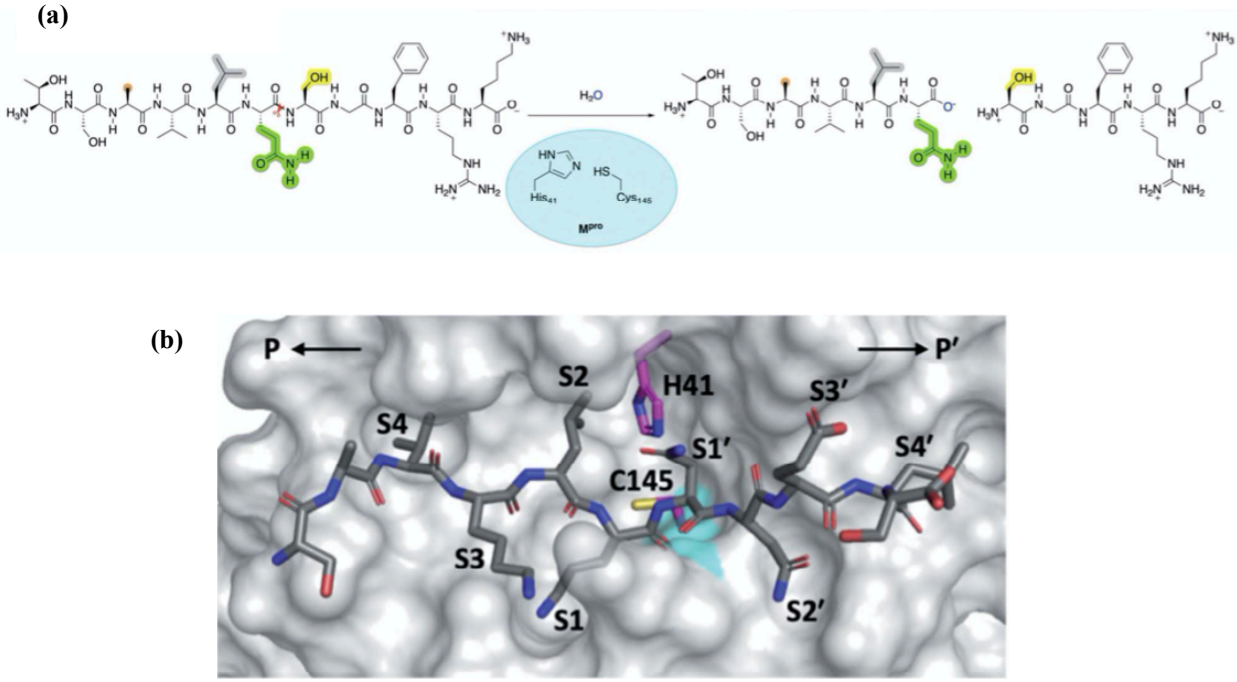


Figure 4.2: (a) The catalyzed reaction of M^{pro} shown by s01. The important substrate residues are highlighted. (b) The binding site of SARS-CoV-2 main protease created using an energy minimized model of *apo* M^{pro} (PDB: 6yb7, light grey surface) of M^{pro} complexed with s05 (dark green sticks). Subsites S4-S4′ are labelled, and the oxyanion hole formed by the M^{pro} backbone NHs (Gly-143, Ser-144, and Cys-145) is shown in cyan. These figures are taken directly from Chan *et al.* (H. T. H. Chan *et al.*, 2021).

I used the S$^{\gamma}$ of Cys-145 in SARS-CoV-2 main protease as the covalent attachment point to design new covalent inhibitors likely to bind to SARS-CoV-2 main protease (Figure 4.3) using my family of c3D-DQN methods, which are described in Section 4.3.1. Henceforth, I will refer to SARS-CoV-2 M^{pro} as "M^{pro}".

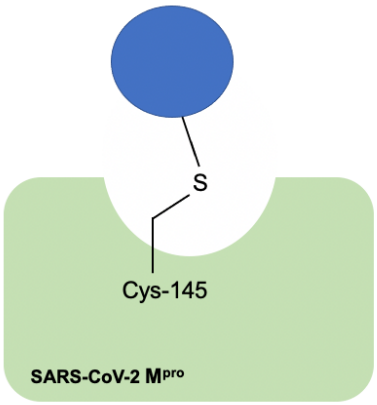


Figure 4.3: Covalent attachment point (S^{γ} of Cys-145) for new covalent inhibitors (blue circle) likely to bind to SARS-CoV-2 main protease (light green).

4.2.3 M^{pro} Dataset from XChem's Fragalysis

XChem's Fragalysis (Douangamath *et al.*, 2020) is a cloud-based platform that has reported 587 X-ray crystal structures for SARS-CoV-2 main protease (M^{pro}) as of September 2021. SARS-CoV-2 M^{pro} formed an excellent test bed for constrained docking, since there are many X-ray structures of covalent ligands that bound to Cys-145, as well as it being of critical therapeutic importance.

In order to test the performance of c3D-DQN methods, 17 covalent ligands were selected using the procedure described in Section 5.2. These ligands were used as the desired target molecule to test various c3D-DQN methods.

4.2.4 Project Aim

In this chapter, I introduce a new docking pipeline of c3D-DQN methods (Section 4.3.1) that are able to perform constrained docking using AutoDock Vina (Trott and Olson, 2010), instead of

commercial docking toolkits such as GOLD (Jones *et al.*, 1997) and Schrödinger's GLIDE (Friesner *et al.*, 2004). The reward function considered in the c3D-DQN methods is the predicted ligand efficiency (Section 3.2.5).

I investigate how the incorporation of constrained docking affects *de novo* molecular designs using DQN. The results are presented in Chapter 6 and a summary of the c3D-DQN methods that I developed are shown in Table 4.2. The c3D-DQN methods include three new methods (c3D-MolDQN, c3D-RxnDQN, and c3D-RxnMolDQN) with each having been tested using two reward functions (LE and LE + T).

Table 4.2: Summary of *de novo* methods developed for constrained/covalent c3D-DQN methods, and which action spaces and reactions were used, together with their reward functions. All methods noted here were developed by the author. LE and T stand for predicted ligand efficiency and Tanimoto similarity between target and generated molecule, respectively.

Method	Atom/Bond Action Space	Reactions Action Space	Docking	Single Objective Reward Function	Multi-Objective Reward Function
c3D-MolDQN	Yes	No	Yes (Constrained)	LE	LE + T
c3D-RxnDQN	No	Yes		LE	LE + T
c3D-RxnMolDQN	Yes	Yes		LE	LE + T

4.3 Methodology

4.3.1 Constrained Docking in Deep Q-Networks (c3D-DQN)

As an extension to the nc3D-DQN methods described in Section 3.3.1, I have also developed structure-based *de novo* methods that use constrained/covalent docking called the c3D-DQN methods. The procedure for the constrained docking methods, c3D-DQN (Figure 4.4), is similar to that of the non-constrained versions, nc3D-DQN (Figure 3.3), but differs by first generating

the compound-amino acid covalent adduct of the newly designed ligand (Figure 4.5(b)) using RDKit via an appropriate SMARTS reaction (Appendix A, Table A3).

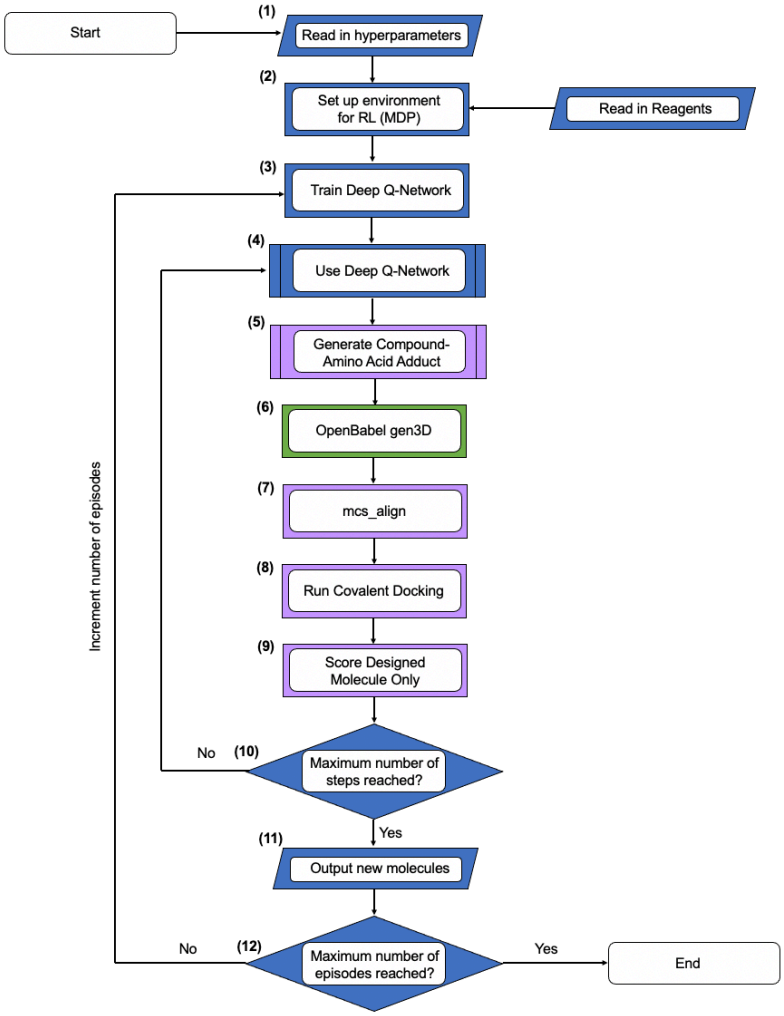


Figure 4.4: Flowchart for c3D-DQN methods (c3D-MolDQN, c3D-RxnDQN, and c3D-RxnMolDQN), where common workflow as nc3D-DQN methods is highlighted in green and unique workflow for c3D-DQN methods is highlighted in purple. Details for the steps in blue can be found in Figure 2.6.

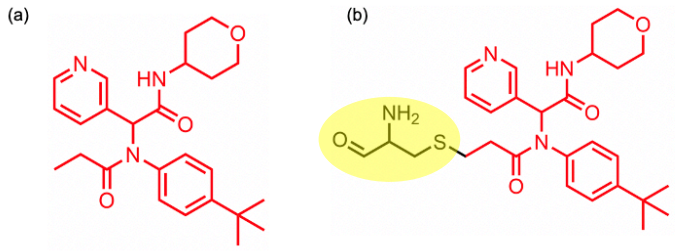


Figure 4.5: The newly generated molecule from DQN in step **(4)** in Figure 4.4 shown in (a) and the compound-amino acid adduct after reacting with cysteine residue in step **(5)** in Figure 4.4 shown in (b) using reaction SMARTS for acrylamide from Appendix A, Table A3. The common substructure between (a) and (b) is highlighted in red. The cysteine that is covalently attached to the protein is highlighted in yellow. (a) is purposely without a double bond for Michael addition in acrylamide since the end-product of c3D-DQN after step **(9)** is (a).

Currently, step **(5)** is tailored towards reacting with cysteine residues (and SARS-CoV-2 main protease), but could be applied to other protein targets if the appropriate SMARTS reaction is supplied. Next in step **(6)**, a 3D-conformation of the compound-amino acid adduct is generated (using OpenBabel's `gen3D` method) in the form of an SDF file. The mcs_align python script (H. T. H. Chan *et al.*, 2021; M. A. Moesser, 2022) is used in step **(7)** to align in 3D the cysteine residue present in the compound-amino acid adduct to the 3D coordinates of the previously prepared cysteine residue in the target protein PDBQT file (*i.e.*, Cys-145 in SARS-CoV-2 main protease), using the maximum common substructure matching algorithm in RDKit (Landrum, 2019). This results in an aligned compound-amino acid adduct SDF file that is used for covalent docking the Cys-145 in SARS-CoV-2 main protease. The command-line version of PyMOL (Schrödinger, 2022) is then used to assign unique atom names to the atoms and convert the aligned compound-amino acid adduct from an SDF file to a PDB file. This assignment is necessary to generate the PDBQT file for docking.

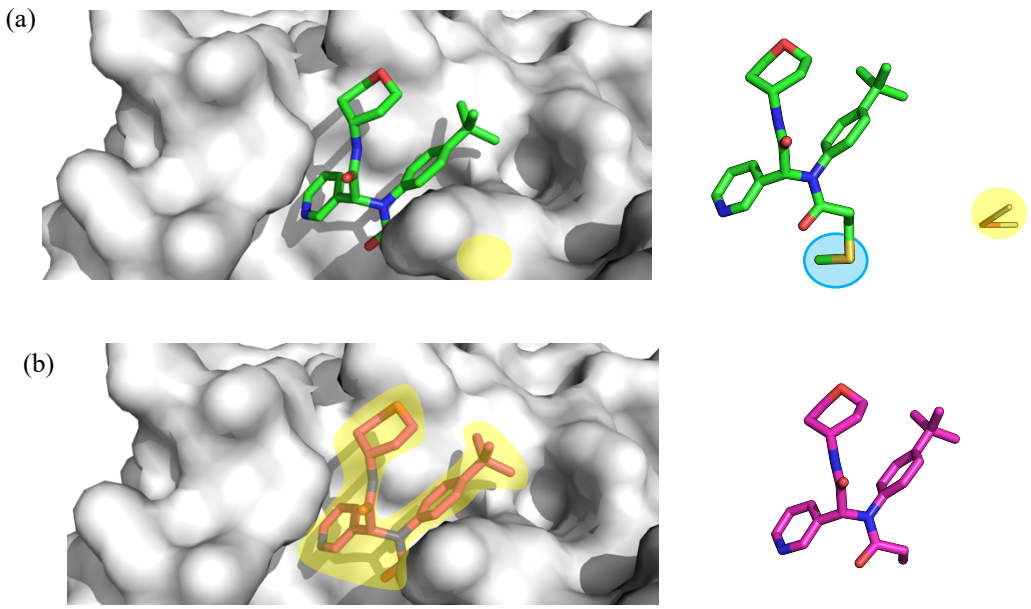


Figure 4.6: (a) The docked pose after performing constrained docking between flexible residue (*i.e.*, prepared at steps **(5)** & **(6)**), H_2O as a ligand (highlighted in yellow), and prepared protein PDBQT file in step **(8)**. The covalent attachment point of Cys-145 is highlighted in light blue. H_2O is the free ligand for AutoDock Vina, which also samples the degrees of freedom of the bound flexible residue. (b) The docked pose from the PDBQT file for score_only docking using AutoDock Vina to obtain the final docking score in step **(9)**. By default, AutoDock Vina computes the docking score of the free ligand with the protein, *i.e.*, H_2O; but for my purposes, I need the score for the covalently bound *de novo* ligand, which requires a second score only operation with AutoDock Vina. These energy calculations are highlighted in yellow.

Afterwards, the aligned compound amino acid adduct PDB file is converted into a PDBQT format using *prepare_ligand4.py* as part of AutoDockTools (G. M. Morris *et al.*, 2009). After aligning the cysteine atoms in the adduct to the Cys-145 in M^{pro}, this covalent adduct PDBQT file is then used as the input flexible residue in the protein together with a previously prepared H_2O PDBQT file, which is used as the free ligand in an AutoDock Vina docking in step **(8)** to the previously prepared protein PDBQT file. The water molecule is used as a ligand to be docked since it acts as a trick to force AutoDock Vina to sample the degrees of freedom of the bound ligand (*i.e.*, flexible residue). The best docked pose of the water plus covalently-modified flexible residue is output in AutoDock Vina as a new PDBQT file (Figure 4.6(a)). This PDBQT file from step **(8)** is used to

perform the second score_only docking with M^{pro} in step **(9)** (Figure 4.6(b)) after removing the covalent attachment point. The docking score from step **(9)** is used to compute for predicted ligand efficiency (Section 3.2.5).

4.3.2 Comparison to nc3D-DQN Methods

The reward functions for both single-objective (Equation 3.13) and multi-objective (Equation 3.14) optimizations are identical to those of the nc3D-DQN (LE) methods described in Section 3.3.1. The action spaces are identical between nc3D-MolDQN and c3D-MolDQN, nc3D-RxnDQN and c3D-RxnDQN, and nc3D-RxnMolDQN and c3D-RxnMolDQN. Although the action spaces are identical between nc3D-DQN and c3D-DQN methods, the covalent warhead needs to exist in the generated molecule for c3D-DQN methods, so the actions can be considered more restricted. Masking of the covalent warhead is performed using SMARTS (Appendix A, Table A4) to the starting molecule, so that the generated molecule (*e.g.*, Figure 4.5(a)) contains the covalent warhead to form the compound-amino acid adduct (*e.g.*, Figure 4.5(b)).The key difference between nc3D-DQN and c3D-DQN methods is whether constrained docking is performed or not. If the algorithm is to perform constrained docking, then it involves extra procedures as described in Section 4.3.1.

5 Methods of Evaluation for 2D-DQN and 3D-DQN

5.1 Overview

In this chapter, the details for methods of evaluation to test the performance of various 2D-DQN and 3D-DQN (both nc3D-DQN and c3D-DQN) methods introduced in Chapters 2-4 are highlighted. These include dataset used as starting compounds for each method (Section 5.2), exploration of hyperparameters (Section 5.3), computational details such as reagents and hyperparameters (Section 5.4), and methods of assessment (Section 5.5). Table 5.1 shows a summary of the metrics used to assess DQN methods. The results of these metrics are all shown in Chapter 6.

Table 5.1: Metrics used to assess 2D-DQN and 3D-DQN methods in Chapter 6. The details of each metric can be found in the corresponding section number in Section 5.5.

	Metrics	Section Number
1	Validity	5.5.1
2	Synthetic Accessibility	5.5.2
3	Reward	5.5.3
4	Walters' Compound Quality	5.5.4
5	Chemical Space Coverage	5.5.5
6	Silly Walks	5.5.6
7	Rediscovery	5.5.7
8	Optimization Efficiency	5.5.8
9	Uniqueness	5.5.9
10	Computational Efficiency	5.5.10
11	Tanimoto Similarity	5.5.11
12	Normalized SuCOS Distribution	5.5.12
13	Lipinski's Rule of Five	5.5.13
14	Multi-objective Optimization	5.5.14
15	Comparison against Published Methods	5.5.15

5.2 Dataset for Method Validations

This section describes the workflow to select the starting compounds to compare the performance of various 2D-DQN (Figure 5.1), nc3D-DQN (Figure 5.2), and c3D-DQN methods (Figure 5.3).

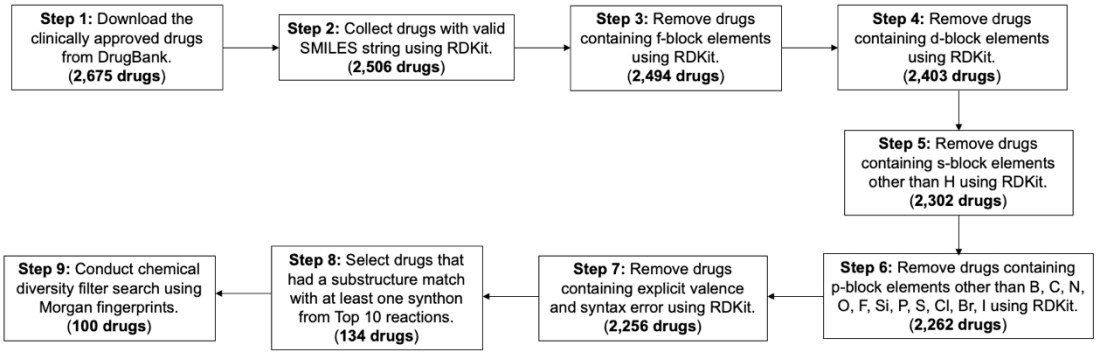


Figure 5.1: Workflow to select 100 chemically diverse drugs from DrugBank (Wishart *et al.*, 2018) using RDKit (Landrum, 2019) to test 2D-DQN methods (MolDQN, RxnDQN, and RxnMolDQN) where the starting molecule is non-empty.

For 2D-DQN methods (Sections 2.3.1-2.3.3), 100 clinically approved drugs from 2,675 drugs in DrugBank (Wishart *et al.*, 2018) were selected using the procedures described in Figure 5.1. These drugs (Appendix B, Figure B2) were used as the starting compound for methods 5-24 (Table 5.2) together with their corresponding baselines. Steps 2 to 7 were performed to select small molecule drugs with chemically valid SMILES string. Step 8 was performed to ensure that at least one of the top 10 reactions (Roughley and Jordan, 2011) can be performed for RxnDQN. Step 9 was computed to select the most chemically diverse set of 100 clinically approved drugs.

For nc3D-DQN (LE) methods (Figure 3.3 in Section 3.3.1), 51 ligand-pairs from the Malhotra set (Malhotra and Karanicolas, 2017) were chosen based on the procedures described in Figure 5.2 (up to Step 6). Step 2 was conducted using RDKit (Landrum, 2019) to ensure that the smaller ligand from the Malhotra set has the capacity to grow up to its larger ligand counterpart. Steps 3a and 3b were implemented for the same reason as Step 8 in Figure 5.1, but this time for nc3D-RxnDQN. Step 5 was performed to avoid unsuccessful docking from AutoDock Vina (Trott and Olson, 2010), which is not parameterized for boron-containing ligands. Similarly, Step 6 was enforced to avoid generation of boron-containing products for nc3D-RxnDQN and nc3D-

RxnMolDQN. The selected 51 smaller ligands used as the starting compound are shown in Appendix B, Figure B5.

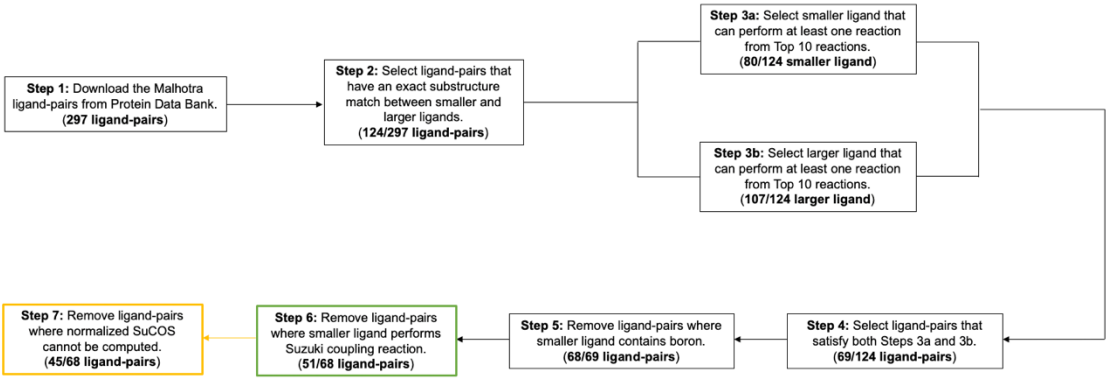


Figure 5.2: Workflow to select 51 ligand-pairs (up to Step 6 in green) and 45 ligand-pairs (up to Step 7 in yellow) from the Malhotra set (Malhotra and Karanicolas, 2017) using RDKit (Landrum, 2019) to test nc3D-DQN (LE) and nc3D-DQN (nSuCOS) methods, respectively, where the starting molecule is non-empty. Step 7 was added for nc3D-DQN (nSuCOS) methods to ensure that normalized SuCOS can be computed for all starting molecules. Each nc3D-DQN method with varying reward function consists of nc3D-MolDQN, nc3D-RxnDQN, and nc3D-RxnMolDQN.

In the case of nc3D-DQN (nSuCOS) methods (Figure 3.4 in Section 3.3.1), the procedures are almost identical to nc3D-DQN (LE) methods except for Step 7. This additional step filters out the receptor of the smaller ligands (*i.e.*, 2v3d, 2y54, 3ati, 3ch9, 3iaf, and 4a23) where the normalized SuCOS (Leung *et al.*, 2019) cannot be computed between the smaller and larger ligands due to poor/no shape and feature overlaps. The selected 45 smaller ligands for the starting compound are listed in Appendix B, Figure B10.

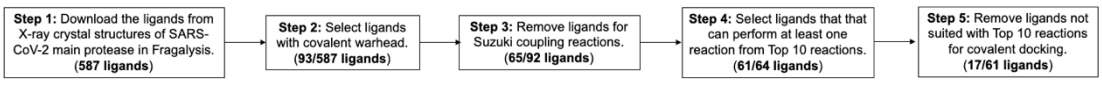


Figure 5.3: Workflow to select 17 covalent ligands from the Fragalysis database of SARS-CoV-2 main protease crystal structures (Douangamath *et al.*, 2020) using RDKit (Landrum, 2019) to test c3D-DQN methods (c3D-MolDQN, c3D-RxnDQN, and c3D-RxnMolDQN).

For c3D-DQN methods (Figure 4.4 in Section 4.3.1), 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) were selected using the procedures in Figure 5.3. Step 2 ensures that a covalent warhead (either acrylamide, chloroacetamide, nitrile, or vinylsulfonamide) exists, which is necessary to perform covalent docking. Step 3 is performed for the same reason as Step 6 in Figure 5.2, but this time for c3D-RxnDQN and c3D-RxnMolDQN. Step 4 is performed for the same reason as Step 8 in Figure 5.1, but this time for c3D-RxnDQN. Step 5 eliminates ligands that are not suited for covalent docking. For example, a ligand that has only one reactive center, which is the covalent warhead so no ligand-design reactions can be performed. The selected 17 covalent ligands are shown in Appendix B, Figure B8.

5.3 Exploration of Hyperparameters

This section shows the hyperparameters that were used to test the performance of 2D-DQN (Table 5.2), nc3D-DQN (Table 5.3), and c3D-DQN methods (Table 5.4).

For 2D-DQN methods, the corresponding baseline methods associated with methods 1-24 in Table 5.2 are defined by random walk, greedy, and ε-greedy (Zhou *et al.*, 2019). At each step in an episode, "random walk" chooses the random action. "Greedy" chooses the action that results in the maximum reward at each step. The difference between "greedy" and "ε-greedy" is the probability in random policy, which is defined as ε for ε-greedy and 1 - ε for greedy. For instance, if $\varepsilon = 0.1$ for ε-greedy policy, then the probability for greedy policy is 0.9. Therefore, ε-greedy is a mix of random walk and greedy. These baselines were used to compare the performance with DQN-based methods.

Table 5.2: List of 2D-DQN methods together with its combination of number of reactions, starting molecule, type of DQN, reagent selection, and Tabu search. Note: 10 reactions for reaction-based action space since ether and ester reactions are counted as 2 reactions (Figure 2.7).

Method Number	Method	Number of Reactions	Starting Molecule[1]	Type of DQN	Reagent Selection[2]	Tabu Search Allowed
1	MolDQN	N/A	empty	naïve	N/A	on
2	MolDQN	N/A	empty	bootstrap	N/A	on
3	MolDQN	N/A	empty	naïve	N/A	off
4	MolDQN	N/A	empty	bootstrap	N/A	off
5	MolDQN	N/A	specific	naïve	N/A	on
6	MolDQN	N/A	specific	bootstrap	N/A	on
7	MolDQN	N/A	specific	naïve	N/A	off
8	MolDQN	N/A	specific	bootstrap	N/A	off
9	RxnDQN	10	specific	naïve	random	on
10	RxnDQN	10	specific	bootstrap	random	on
11	RxnDQN	10	specific	naïve	random	off
12	RxnDQN	10	specific	bootstrap	random	off
13	RxnDQN	10	specific	naïve	diverse	on
14	RxnDQN	10	specific	bootstrap	diverse	on
15	RxnDQN	10	specific	naïve	diverse	off
16	RxnDQN	10	specific	bootstrap	diverse	off
17	RxnMolDQN	10	specific	naïve	random	on
18	RxnMolDQN	10	specific	bootstrap	random	on
19	RxnMolDQN	10	specific	naïve	random	off
20	RxnMolDQN	10	specific	bootstrap	random	off
21	RxnMolDQN	10	specific	naïve	diverse	on
22	RxnMolDQN	10	specific	bootstrap	diverse	on
23	RxnMolDQN	10	specific	naïve	diverse	off
24	RxnMolDQN	10	specific	bootstrap	diverse	off

[1] "Specific" refers to 100 clinically approved drugs from DrugBank chosen based on Figure 5.1.

[2] "Random" selects 50 compounds from reagent set including seed reagents for each reaction (*e.g.*, 10,487 reagents for amine reagents), and in a typical run, there may be up to 40 actions; "Diverse" selects 50 random reagents from most chemically diverse reagent set based on RDKit's ECFP4 fingerprint with radius 2 and 2,048 bits including seed reagents (*e.g.,* 98 reagents for amine reagents).

Number of reactions, starting molecule, type of DQN, and tabu search are all described in Sections 2.3.1, 2.3.2, and 2.3.4. Two types of reagent selection ("Random" or "Diverse") were used to compare the performance between reagents sets. The difference between this reagent selection is whether chemically diverse reagent set is considered (*i.e.*, "Diverse") or not (*i.e.*, "Random"). The reagent set used for "Random" selection is called "All Reaxys" set, which consists of all the corresponding reagents from Reaxys (Elsevier, 2020) for each synthon in the top 10 reactions (Roughley and Jordan, 2011) including a "seed" reagent derived from the target molecule. Seed reagents were also added to the Reaxys reagents to ensure that the algorithm has the capacity to

recover the initial starting molecule from DrugBank (Wishart *et al.*, 2018). Moreover, the reagent set used for "Diverse" selection is called "Selected Reaxys" set, which includes 50 chemically diverse compounds that are selected at random from all Reaxys reagents for each synthon in a reaction with seed reagents. There were 77,710 reagents in total for Reaxys reagents with seed reagents.

Table 5.3: List of nc3D-DQN methods together with the combination of number of reactions, starting molecule, type of DQN, reagent selection, and if Tabu search was used. nc3D-DQN methods include both nc3D-DQN (LE) and nc3D-DQN (nSuCOS) methods.

Method	Number of Reactions	Starting Molecule[1]	Type of DQN	Reagent Selection[2]	Tabu Search Allowed
nc3D-MolDQN	0	specific	naïve	random	on
nc3D-RxnDQN	9	specific	naive	random	on
nc3D-RxnMolDQN	9	specific	naïve	random	on

[1] "Specific" refers to 51 or 45 smaller ligands from the Malhotra set chosen based on Figure 5.2.

[2] "Random" selects 50 compounds from reagent set including seed reagents for each reaction (*e.g.,* 10,079 reagents for amine reagents), and in a typical run, there may be up to 10 actions.

For nc3D-DQN methods (Table 5.3), only nine reactions of Roughley *et al.*'s top 10 reactions are included in nc3D-RxnDQN and nc3D-RxnMolDQN to avoid generation of boron-containing molecules from the Suzuki coupling reaction. "Specific" starting molecules were chosen from the Malhotra set (Section 3.2.6). The remaining hyperparameters were selected based on the results from the 2D-DQN methods. "Naïve" DQN (Section 2.3.1) was chosen since when comparing the performance of the 2D-DQN methods, naïve DQN methods performed better than bootstrap DQN methods. "Random" reagent selection was used since for the 2D-DQN methods, the results did not significantly differ between random and diverse reagent selections. Tabu search was turned "on" since based on the results from the 2D-DQN methods, the "on" models generally resulted in shorter computation time.

For c3D-DQN methods (Table 5.4), the hyperparameters except for reagent selection were chosen for the same reason as nc3D-DQN methods (Table 5.3). For reagent selection, the c3D-DQN methods did not contain the "seed" reagents, which would have included the possibility for rediscovery of target ligands.

Table 5.4: List of c3D-DQN methods together with its combination of number of reactions, starting molecule, type of DQN, reagent selection, Tabu search, and seed reagents.

Method	Number of Reactions	Starting Molecule[1]	Type of DQN	Reagent Selection[2]	Tabu Search Allowed	Seed Reagents
c3D-MolDQN	0	specific	naïve	random	on	no
c3D-RxnDQN	9	specific	naive	random	on	no
c3D-RxnMolDQN	9	specific	naïve	random	on	no

[1] "Specific" refers to 17 covalent ligands for SARS-CoV-2 M^{pro} chosen based on Figure 5.3.

[2] "Random" selects 50 compounds from reagent set for each reaction (*e.g.,* 10,060 reagents for amine reagents), and in a typical run, there may be up to 10 actions.

5.4 Computational Details for Method Validations

5.4.1 Reagents in Reaction-based Action Space

For 2D-DQN methods (Table 5.2), I used reagents downloaded from Reaxys (Wishart *et al.*, 2018) as described in Section 2.2.12. In particular, I used 50 randomly selected reagents for each reaction in a particular synthon. For example, for the amine synthon in the urea formation reaction, there are two possible reactions: one with a primary amine and the other with a secondary amine. In this case, I used 50 randomly selected primary amine reagents and 50 randomly selected secondary amines for the urea reaction.

For nc3D-DQN methods (Table 5.3), reagents from Reaxys were also used. However, I filtered out reagents containing boron atoms, since boron parameters are not included in AutoDock Vina (Trott and Olson, 2010) and results in a binding affinity score of zero. The number of reagents for

each reaction after applying this filter is reported in Appendix B, Table B4. The random selection of reagents for each reaction is like the 2D-DQN methods.

For c3D-DQN methods (Table 5.4), reagents from Reaxys were filtered for boron atoms for the same reason as nc3D-DQN methods. The number of reagents after applying this filter is reported in Appendix B, Table B7. The random selection of reagents is also used.

5.4.2 Hyperparameters (JSON)

For 2D-DQN methods (Table 5.2), I used the original default settings used by MolDQN (Zhou *et al.*, 2019) for hyperparameters defined in the JSON files except for the number of steps, episodes, and actions; 3,000 episodes were used to test the methods and their corresponding baselines. Originally, 5,000 episodes were used for MolDQN. However, there was no significant improvement in reward between 3,000 to 5,000 episodes, so I decided to run 2D-DQN methods for fewer episodes to reduce computational cost. The actions are defined by the action space as explained in Sections 2.3.1-2.3.3.

In order to determine the appropriate number of steps for RxnDQN and RxnMolDQN, the reagents from the "All Reaxys" set and "Selected Reaxys" set (Section 5.3) were combined separately (Appendix B, Table B1). When plotting the distribution of number of atoms for each reagent in the "All Reaxys" set, 88% were found to have fewer than 40 atoms and the average number of atoms was found to be approximately 17 atoms. When assuming that most drug-like molecules contain fewer than 40 atoms, at least three successive reaction steps would be required for the reaction-based action space to reach structures this size. Therefore, 5 steps were chosen

for RxnDQN, and 10 steps were chosen for RxnMolDQN. For MolDQN, the default of 40 steps was used.

For nc3D-DQN methods (Table 5.3), the original default settings for MolDQN were used except for the number of steps and episodes. For nc3D-RxnDQN, I set to 5 steps and 300 episodes. For nc3D-MolDQN and nc3D-RxnMolDQN, I used 10 steps and 300 episodes. I chose the same number of steps for reaction-based nc3D-DQN methods as their 2D counterparts, since we prefer the algorithm to generate the optimized molecule in as few synthetic steps as possible. In the case of nc3D-MolDQN, I chose 10 steps (rather than 40 steps in 2D-DQN methods) since docking is more computationally expensive than operating on SMILES strings. I used 300 episodes, since based on my previous tests with nc3D-DQN methods (both LE and nSuCOS), there was no significant improvement in the reward when I ran for longer episodes. Therefore, I prioritized the computational efficiency for these nc3D-DQN methods so that each run could be performed within a few hours using the University of Oxford's Advanced Research Computing (ARC) resource (Richards, 2015).

For c3D-DQN methods (Table 5.4), the hyperparameters were identical to nc3D-DQN methods for the same reason.

5.4.3 Molecular Weight Filter

For 2D-DQN methods (Table 5.2), I decided to place a molecular weight filter for the new molecules generated in RxnDQN and RxnMolDQN (molecular weight $\leq$ 750 Da) to prevent the molecules from growing too large. This filter was based on the HIV-1 protease inhibitor with the highest molecular weight, ritonavir at 720.9 Da (Kim *et al.*, 2021). This drug was used as a

reference since the crystal structures are readily available in PDB (Berman *et al.*, 2000) and protease inhibitors are well-studied targets for drug discovery.

For 3D-DQN methods (*i.e.*, both nc3D-DQN and c3D-DQN methods, Tables 5.3.-5.4), I placed a molecular weight filter of 500 Da to prevent the molecules from growing too large by performing docking (Sections 3.3.1 and 4.3.1). This filter was selected to comply with Lipinski's "rule" of five, which is a set of criteria that is often satisfied by orally active drugs and includes: 1) no more than 5 hydrogen bond donors; 2) no more than 10 hydrogen bond acceptors; 3) molecular mass less than 500 Da; and 4) logP does not exceed 5 (Lipinski *et al.*, 2001).

5.4.4 AutoDock Vina Details

For nc3D-DQN (LE) methods (Table 5.3), I pre-processed the proteins containing the smaller ligands from the Malhotra set using PyMOL (Schrödinger, 2022) to perform the docking using AutoDock Vina (Trott and Olson, 2010). These pre-processed proteins were used to perform the docking of the generated molecules (*i.e.*, ligands) at each step in an episode (Section 3.3.1). After downloading the biological assembly of smaller ligand's protein from the PDB (Berman *et al.*, 2000) in PDB format, PyMOL was used to align each protein and then separate the respective ligand from the respective protein so that it could later be converted to PDBQT format. I used *prepare_ligand4.py* and *prepare_receptor4.py* from AutoDockTools (G. M. Morris *et al.*, 2009) to convert processed proteins' and ligands' PDB files from PyMOL, respectively, into PDBQT files. To compute the binding affinity in AutoDock Vina, the user needs to specify parameters: grid center and size, ligand and protein files, and exhaustiveness. The ligand file preparation is described in Section 3.3.1. ADT was employed to determine the grid center and size for the dockings. The grid size for x, y, and z directions were mostly set to all 20 Å. For proteins with

smaller binding pocket (*i.e.*, 3s75 and 3s76), I used 15 Å for the *x*, *y*, and *z* directions. Since AutoDock Vina was used, I set the grid spacing to 1 Å when determining the grid center and size. I chose 1 for search exhaustiveness to increase speed.

For nc3D-DQN (nSuCOS) methods (Table 5.3), I pre-processed each protein-pair (*i.e.*, the complexes containing the larger and smaller ligands) from the Malhotra set to perform the docking using AutoDock Vina. These pre-processed proteins from the smaller ligand-complexes were docked in a similar way as nc3D-DQN (LE) methods. After downloading the biological assembly of each protein-pair from the PDB (PDB format), I used PyMOL to align the protein in the smaller ligand-protein complex to the larger ligand protein-complex. These are then separated into respective proteins, which are then separated into respective ligands so that both protein-ligand pairs could later be converted to PDBQT format. The remaining procedures for PDBQT file preparation and setting parameters for AutoDock Vina are identical to nc3D-DQN (LE) methods. In order to prepare the reference SDF for normalized SuCOS, I converted the larger ligands' PDBQT files into SDF files using OpenBabel's `gen3D` (O'Boyle *et al.*, 2011).

For c3D-DQN methods (Table 5.4), PDB-formatted files of the desolvated apo protein (*i.e.*, xxx_apo_desolv.pdb) and ligand (*i.e.*, xxx.pdb) where xxx is the name of the SARS-CoV-2 M^{pro} X-ray crystal structure (*e.g.*, Mpro-x3117_0A) from Fragalysis (Douangamath *et al.*, 2020) were downloaded. These proteins were used as the receptor to perform the docking of the *de novo* designed molecules (*i.e.,* ligands) at each step in an episode (Section 4.3.1). The procedures for PDBQT file preparation and setting parameters for AutoDock Vina are identical to nc3D-DQN (LE) methods except that the grid dimensions are set to all 25 Å.

5.5 Methods of Assessment

This section describes the metrics that were used to assess the performance of 2D-DQN (Table 5.2), nc3D-DQN (Table 5.3), and c3D-DQN (Table 5.4) methods. Unless stated otherwise, the metrics are consistent between 2D-DQN and 3D-DQN methods (Figure 1.17).

5.5.1 Validity

Validity is a metric that determines whether the SMILES strings generated by each method are chemically valid according to RDKit (Landrum, 2019). Chemically valid SMILES strings are chemical structures without any RDKit errors. This value ensures that the algorithm generates chemically correct molecules. Once we have determined the number of valid SMILES strings for each method, the percentage of validity is given by:

$$P_{valid} = \left(\frac{Number\ of\ valid\ SMILES\ strings}{Number\ of\ generated\ SMILES\ strings}\right) \times\ 100\% \qquad (5.1)$$

where the number of generated SMILES strings correspond to the number of episodes for DQN methods. Therefore, the denominator for P_{valid} is 3,000 for 2D-DQN methods and 300 for 3D-DQN methods.

5.5.2 Synthetic Accessibility

Synthetic accessibility (SAscore) was calculated using the method described in Section 2.2.10. For 2D-DQN methods (Table 5.2), I computed the SAscore for molecules generated over 3,000 episodes for each run of every DQN method. For methods 5-24 and their corresponding baselines (random walk, greedy, and ε-greedy) in Table 5.2, I calculated the average and standard deviation of SAscore for the 100 selected DrugBank drugs for each method (Wishart *et al.*, 2018). For 3D-DQN methods (Tables 5.3-5.4), the average and standard deviation of SAscore were computed

for Malhotra smaller ligands (nc3D-DQN methods) and SARS-CoV-2 M^{pro} (c3D-DQN methods) using 300 generated molecules for each run of every method.

5.5.3 Reward

For both 2D-DQN and 3D-DQN methods, the molecules with top three highest reward values together with its corresponding averages and overall average were determined for each method. In the case of nc3D-DQN (LE) and c3D-DQN methods (Tables 5.3-5.4), the average and standard deviation of top 100 reward values were also computed for each method.

5.5.4 Walters' Compound Quality

Walters' compound quality (Walters, 2018) attempts to describe a compound's druglikeness by applying structural filters and counting how many RDKit property filters a compound passes. It ranges from 0 to 1, and rates the generated compounds' "quality" for each method based on hydrogen bond acceptors/donors, logP, molecular weight, topological polar surface area, and rules such as SureChEMBL, Glaxo, and PAINS from ChEMBL (Bento *et al.*, 2014) and in-house rule collections (N. Brown *et al.*, 2019), where achieving a higher positive value is more favorable. For 2D-DQN methods (Table 5.2), the score is reported as the value itself for methods 1-4 and their corresponding baselines. For methods 5-24 and their corresponding baselines, the average and standard deviation is reported across DrugBank drugs (Wishart *et al.*, 2018), for each method. For 3D-DQN methods (Tables 5.3-5.4), the average and standard deviation from Malhotra smaller ligands (nc3D-DQN methods) and SARS-CoV-2 M^{pro} (c3D-DQN methods) are reported.

5.5.5 Chemical Space Coverage

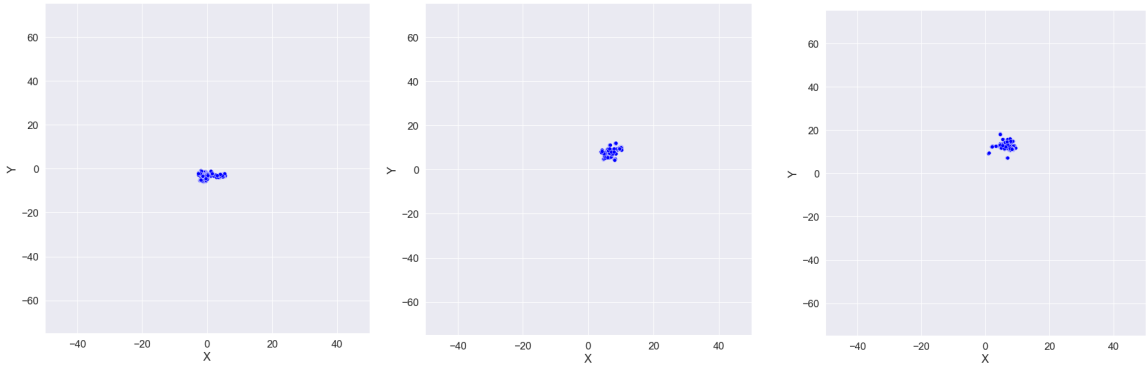
I used the Uniform Manifold Approximation and Projection (UMAP) to estimate the chemical space coverage (McInnes *et al.*, 2018). UMAP performs non-linear dimensionality reduction and can be used for visualization by using the nearest neighbor graph. Nearest neighbor graph is used to construct the initial high-dimensional graph. As the number of nearest neighbors increases, UMAP connects more and more neighboring points when constructing the graph representation of the high dimensional data, which leads to projection that more accurately depicts the global structure of the data. The chief advantages of using UMAP over t-distributed Stochastic Neighbor Embedding (t-SNE) (van der Maaten and Hinton, 2008), another popular dimensionality reduction method, is that UMAP scales well with increasing sample size, and takes into account both cluster distances as well as similarities, attempting to preserve a fuzzy topological surface. In the case of t-SNE, the cluster similarities are not guaranteed.

For 2D-DQN methods (Table 5.2), all the 3,000 molecules generated after each episode for each starting molecule and each method were converted into Morgan fingerprints (Rogers and Hahn, 2010) with a length of 2,048 bits and a radius of 2 (Section 2.3.1). The results from the UMAP dimensionality reduction of these fingerprints were visualized using Seaborn (Waskom, 2021) and the coordinates were stored into Pandas (McKinney, 2010) dataframes. I then analyzed the chemical space coverage of the same starting molecule for different methods. I concatenated all the molecules generated from the same starting molecule across the different methods into a single Pandas dataframe, so that each UMAP projection is the same for all subsets and the UMAP grid represents the same chemical space.

After representing each method's occupied chemical space in a scatterplot where the example is shown in Figure 5.4(a), I conducted an image quantization to determine the area of coverage for each method with the same starting molecule using the code shown in Figure 5.4(b). The proportion of chemical space coverage was computed by:

$$P_{chemical\ space} = \left(\frac{Area\ of\ Pixels\ for\ Method\ with\ Starting\ Molecule\ A}{Area\ of\ Pixels\ for\ Method\ with\ Largest\ Area\ for\ Starting\ Molecule\ A}\right) \times 100\% \quad (5.2)$$

(a)



(b)

```
pixel_result = []
for input_file_name in os.listdir(working_directory):
    sep = 'drugbank_'
    rest = input_file_name.split(sep, 1)[1]
    sep1 =  '_' + reward
    drug_name = rest.split(sep1, 1)[0] #drug number
    sep2 = '_drugbank'
    method_name = input_file_name.split(sep2, 1)[0] # method name
    image = np.array(Image.open(working_directory+'/'+input_file_name).convert('RGB'))
    # Work out what we are looking for (blue)
    sought = [0,0,255]
    # Find all pixels where the 3 RGB values match "sought", and count them
    result = np.count_nonzero(np.all(image==sought,axis=2))
    pixel_result.append((drug_name, method_name, result))
```

Figure 5.4: Procedure to compute relative chemical space coverage using UMAP by (a) creating a scatter plot for each experiment starting molecule for a method (*i.e.*, left – MolDQN, center – RxnDQN, and right – RxnMolDQN with its corresponding relative chemical space coverage out of 100 shown at the bottom of each plot) after concatenating all the molecules generated from a same starting molecule into a single Pandas dataframe, (b) image quantization code to recognize the pixel sizes for blue data points, which gets used later to determine the relative chemical space coverage between different methods for the same starting molecule. The grid size for these scatter plots were determined after visualizing all scatter plots generated for 2D-DQN methods. Overplotting of data points does not matter in (a) since we want to find the overall chemical space coverage by only considering unique molecules.

The chemical space coverage was computed in the similar manner for 3D-DQN methods (Tables 5.3-5.4), but each starting molecule for each method generated 300 molecules instead of 3,000 molecules.

5.5.6 Silly Walks

Patrick Walters' silly walks measures the degree of "silliness" for generated molecules for each method and ranges from a score of 0 (least silly molecule, most favorable) to 1 (most silly molecule, least favorable) by comparing the occurrence of unusual fingerprint bits against reference drug molecules from ChEMBL (computed using Morgan Fingerprints with radius 2) as defined in PatWalters' GitHub repository (Walters, 2021). For both 2D-DQN and 3D-DQN methods, the average and standard deviation for each method is reported.

5.5.7 Rediscovery

"Rediscovery" determines whether the starting (or target) molecule can be recovered by each method based on InChI keys (Goodman *et al.*, 2021) generated using RDKit (Landrum, 2019). For 2D-DQN methods (Table 5.2), this value checks the algorithm's capacity to recover the initial target drug molecule. For nc3D-DQN methods (Table 5.3), two types of rediscoveries are considered: rediscovery of Malhotra smaller and larger ligands. The rediscovery of smaller ligands determines whether the algorithm has the capacity to regenerate the starting smaller ligand. The rediscovery of larger ligands determines whether the starting molecule, *i.e.*, the smaller ligand from the Malhotra set, can grow and recover the larger ligand from the Malhotra set for the same ligand-pair generated using RDKit. For c3D-DQN methods (Table 5.4), the process for determining ligand rediscovery is like the 2D-DQN methods except that the seed reagent was not used.

5.5.8 Optimization Efficiency

Optimization efficiency monitors the progression of the desired reward function versus the number of episodes for each method. This is to determine the number of episodes needed to reach an optimal reward. Each 2D-DQN, nc3D-DQN (LE) and c3D-DQN, and nc3D-DQN (nSuCOS) methods monitors QED (Section 2.2.11) or penalized logP (Section 2.2.10), predicted ligand efficiency (Section 3.2.5), and normalized SuCOS (Section 3.2.7), respectively. For 2D-DQN methods, all results for reward at each episode (*i.e.*, 100 data points per episode) are considered for each action space. In the case of 3D-DQN methods, the average reward at each episode across starting molecules (*i.e.*, one data point per episode) is considered for each action space.

5.5.9 Uniqueness

"Uniqueness" is the percentage of unique SMILES strings generated by a method as measured by unique InChI keys (Goodman *et al.*, 2021) generated using RDKit (Landrum, 2019). This value measures the degree of molecules' duplication, which can limit the chemical space exploration. Once determined the number of unique SMILES strings for each method, the percentage of uniqueness was found by:

$$P_{uniqueness} = \left(\frac{Number\ of\ unique\ SMILES\ strings}{Number\ of\ generated\ SMILES\ strings}\right) \times\ 100\% \qquad (5.3)$$

where the number of generated SMILES strings correspond to the number of episodes for DQN methods. Therefore, the denominator for $P_{uniqueness}$ is 3,000 for 2D-DQN methods and 300 for 3D-DQN methods. For 2D-DQN methods (Table 5.2), the average percentage uniqueness for first 1,500 molecules in each test case for these methods is found (*i.e.*, denominator in Equation 5.3 is 1,500). In the case of 3D-DQN methods (Tables 5.3-5.4), the average percentage uniqueness for first 150 molecules in each test case for these methods is reported.

5.5.10 Computational Efficiency

Computational efficiency refers to the total time (in hours) it takes for each test run for the DQN methods to complete. The purpose behind this metric is to compare the computational time for various DQN methods. My goal is to achieve useful results for medicinal chemists in less computational time while using the fewest resources possible. The results for this work were obtained using the University of Oxford's ARC supercomputer (Richards, 2015). I submitted all jobs on a single CPU node (CPU: 48 core Cascade Lake (Intel Xeon Platinum 8,268 CPU @ 2.90 GHz); Memory: 392 GB) on ARC with a maximum wall clock time of five days for 2D-DQN methods and two days for 3D-DQN methods. For 2D-DQN methods (Table 5.2), the computation time is reported as the value itself for methods 1-4 and their corresponding baselines, and the average and standard deviation across 100 DrugBank drugs for methods 5-24 and their corresponding baselines. For 3D-DQN methods (Tables 5.3-5.4), the average and standard deviation were obtained for each method using the relevant dataset (Section 5.2).

5.5.11 Tanimoto Similarity

Tanimoto similarity is a popular metric for comparing chemical structures. In general, a Tanimoto similarity value between two molecules greater than 0.85 is considered structurally similar. For 2D-DQN methods (Table 5.2), I converted SMILES strings generated by methods 5-24 and their corresponding baselines (3,000 SMILES strings per starting molecule and method) into Morgan fingerprints (Rogers and Hahn, 2010) with radius 2 and length of 2,048 bits and compared against the corresponding Morgan fingerprint of the 100 DrugBank starting molecules to compute the Tanimoto similarity (Maggiora *et al.*, 2014). This determines how distant the generated molecules are from the initial starting target molecule. Tanimoto similarity for top 1, 10, and 20 of the generated molecules are reported.

In the case of nc3D-DQN methods (Table 5.3), the procedure for obtaining Tanimoto similarity score between the generated molecules (*i.e.*, 300 SMILES strings per starting molecule and method) and the reference molecule is identical to the 2D-DQN methods. However, there are two types of reference molecule for nc3D-DQN methods: Malhotra smaller and larger ligands. The former determines how distant the generated molecules are from the initial starting molecule, whilst the latter determines how distant the generated molecules are from the larger (target) ligand. Tanimoto similarity for top 1, 10, and 20 of the generated molecules are also reported for both smaller and larger ligands.

For c3D-DQN methods (Table 5.4), Tanimoto similarity is computed in a similar way as 2D-DQN methods, but 300 SMILES strings per starting molecule and method.

5.5.12 Normalized SuCOS Distributions

SuCOS is an open-source shape and chemical feature overlap metric, which computes the combined overlap score between two molecules as described in Section 1.10 (Leung *et al.*, 2019). For 3D-DQN methods (Tables 5.3-5.4), I used normalized SuCOS (nSuCOS) to determine the combined overlap score between generated molecules (*i.e.*, 300 SMILES strings per starting molecule and method) and starting molecules (*i.e.*, Malhotra smaller ligands or SARS-CoV-2 M^{pro}) for each method. In general, nSuCOS values greater than 0.55 are equivalent to RMSD values of 2 Å or lower. Percentage of nSuCOS ≥ 0.55 are reported for each 3D-DQN methods.

5.5.13 Lipinski's Rule of Five

For 3D-DQN methods (Tables 5.3-5.4), I computed the proportion of molecules for each method that passed Lipinski's rule of five (Section 5.4.3), which is a popular metric used in drug discovery. This metric is reported based on the average percentage of generated molecules from the relevant dataset (Section 5.2) that satisfied this rule for each method. The corresponding results using the same dataset as 3D-DQN methods are reported for 2D-DQN methods with QED as the reward.

5.5.14 Multi-objective Optimizations

For 2D-DQN methods (Table 5.2), Equation 2.41 was used to perform multi-objective optimization between QED and Tanimoto similarity. Tanimoto similarity was computed between the generated molecules and their corresponding initial DrugBank drug molecule (Figure 5.1). The parameter of ω ranged from 0.0, 0.2, 0.4, 0.6, 0.8 to 1.0. It should be noted that Zhou *et al.* used a Morgan fingerprints (Section 2.3.1) with radius of 3 for this multi-objective reward function, while I have used radius 2 in my analyses. Morgan fingerprints with radius 2 (*i.e.*, ECFP4) and radius 3 (*i.e.*, ECFP6) are commonly used in cheminformatics and both are known to perform well to represent molecular fingerprints (O'Boyle and Sayle, 2016).

For 3D-DQN methods (Tables 5.3-5.4), I conducted a comparison of multi-objective optimization (Sections 3.3.3 and 4.3.2) for nc3D-DQN (LE), nc3D-DQN (nSuCOS), and c3D-DQN methods using reward progression.

5.5.15 Comparison against Published Methods

The results for QED, penalized logP, and validity for 2D-DQN methods are compared to other state-of-the-art techniques including Objective Reinforced Generative Adversarial Networks (ORGAN) (Guimaraes *et al.*, 2017), Junction Tree Variational Autoencoder (JT-VAE) (W. Jin *et al.*, 2018), and Graph Convolutional Policy Network (GCPN) (You *et al.*, 2018). For QED and penalized logP, the first, second, and third best values with its corresponding average and overall average across 3,000 episodes for each DrugBank compound are considered. For validity, it considers the percentage of compounds that are chemically valid according to the RDKit definition (Section 5.5.1).

6 Results and Discussion

6.1 Overview

This chapter shows the results for 2D-DQN (Sections 2.3.1-2.3.3, Table 5.2), nc3D-DQN (Section 3.3.1, Tables 5.3-5.4), and c3D-DQN (Section 4.3.1, Table 5.5) methods using the "Methods of Assessment" described in Section 5.5 with the corresponding dataset from Section 5.2. The summary of the metrics used for evaluating different DQN methods is shown in Table 5.1. Unless stated otherwise, the 2D-DQN methods in this section is referring to methods 5, 15, and 19 for MolDQN, RxnDQN, and RxnMolDQN from Table 5.2, respectively. These methods were chosen as "best methods" for 2D-DQN methods when conducting hyperparameter optimizations. When deciding the "best methods" for 2D-DQN methods, I prioritized for computational efficiency. This is because the results did not significantly vary by changing hyperparameters (Table 5.2) for 2D-DQN methods, and I aim to propose an algorithm that satisfies these metrics (Table 5.1) with the least computational cost. For metrics where both nc3D-DQN methods with reward function of predicted ligand efficiency or normalized SuCOS are considered, I am distinguishing these methods as "nc3D-DQN (LE) methods" and "nc3D-DQN (nSuCOS) methods", respectively.

6.2 Validity

All 2D-DQN (Table 5.2) and 3D-DQN (Tables 5.3-5.4) methods generated molecules that were 100% chemically valid according to RDKit (Landrum, 2019) definition (see Section 5.5.1 for definition). This reaffirmed that DQN methods are constantly able to generate compounds with 100% chemical validity regardless of the types of action space (atom/bond-based, reaction-based, and hybrid-based) and reward functions (QED, penalized logP, LE, and nSuCOS).

For 2D-DQN methods, all results for validity including baselines using QED and penalized logP as a reward function are shown in Appendix D, Tables D1 and D2, respectively. These values were approximated to nearest three decimal places.

6.3 Synthetic Accessibility

Figure 6.1 shows the average synthetic accessibility (Section 5.5.2) for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods with varying reward functions and docking types (if relevant): 2D-DQN (QED), nc3D-DQN (LE), nc3D-DQN (nSuCOS), and c3D-DQN (LE) methods. For MolDQN-based methods (blue box plots), the median of average synthetic accessibility ranges from ~4.4 to 6.7. For reaction-based methods, this value ranges from ~2.8 to 3.2 for RxnDQN-based methods (orange box plots) and ~3.1 to 4.2 for RxnMolDQN-based methods (green box plots). Therefore, reaction-based methods can generate compounds that are more likely to be either synthetically accessible or purchasable than atom/bond-based methods. In addition, reaction-based methods are on average consistently able to generate compounds within the range of catalogue molecules ($2 \leq$ SAscore $\leq$ ~4) regardless of the type of reward functions (*i.e.*, 2D- and 3D-based rewards) and docking (*i.e.*, non-constrained and constrained). These results suggest that the reaction-based methods can be used to generate compounds that are synthetically accessible, which achieves the goal of overcoming one of the limitations of MolDQN-based methods (Section 2.2.14).

On the other hand, for MolDQN-based methods, the median of average synthetic accessibility improves when moving from 2D-based MolDQN (~6.7, Figure 6.1(a)) to 3D-based MolDQN methods (ranges from ~4.4 to 4.9, Figures 6.1(b)-(d)). This implies that the structure-based reward functions of predicted ligand efficiency (Section 3.2.5) and normalized SuCOS (Section 3.2.7)

guide the chemical space exploration to generate compounds that are closer to the catalogue molecules for MolDQN-based methods.

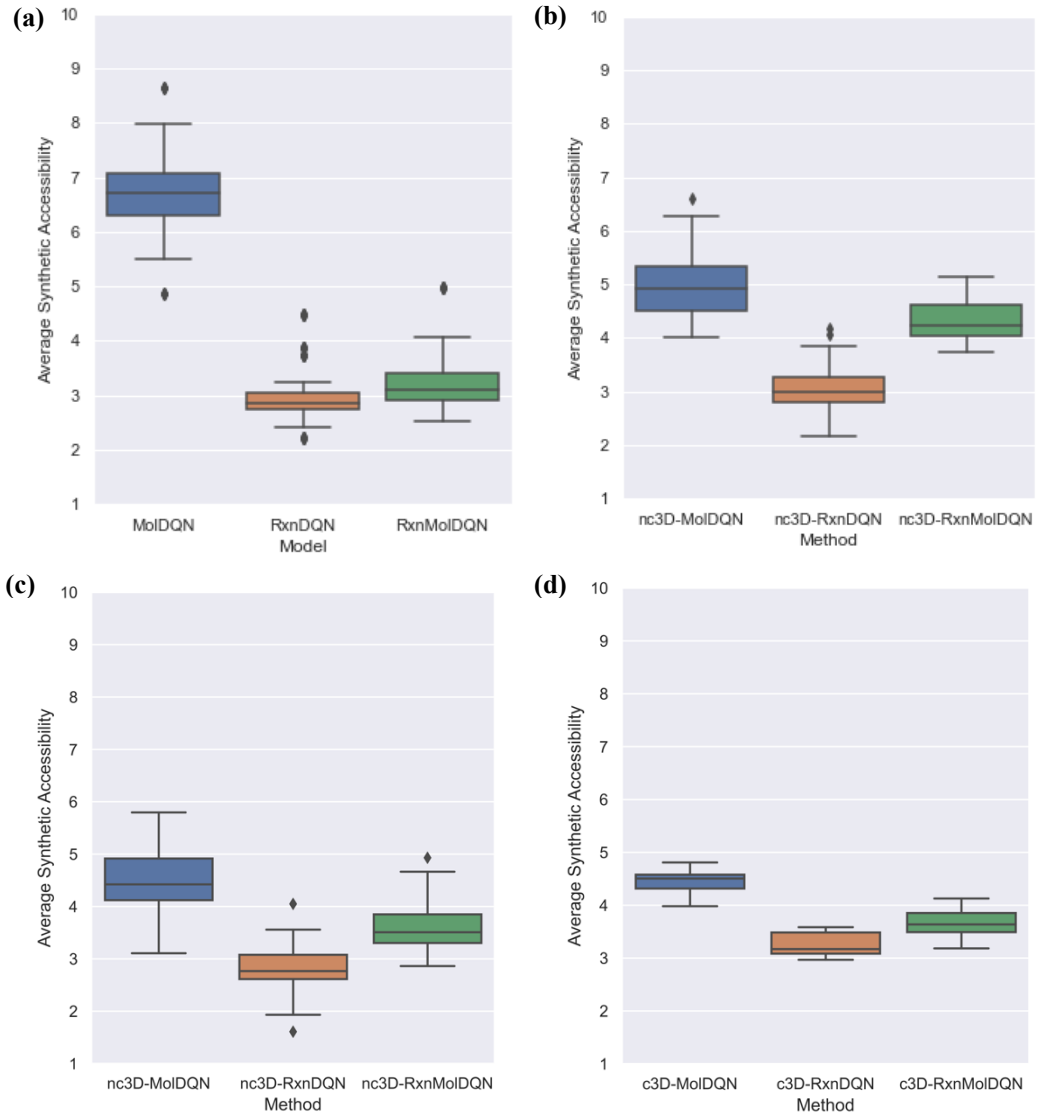


Figure 6.1: Average synthetic accessibility for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods with respective reward function in **(a)** 2D-DQN (QED), **(b)** nc3D-DQN (LE), **(c)** nc3D-DQN (nSuCOS), and **(d)** c3D-DQN (LE). These values are computed using the dataset described in Figure 5.1 for **(a)**, Figure 5.2 for **(b)-(c)**, and Figure 5.3 for **(d)**. The synthetic accessibility improves from 10 (least synthetically accessible) to 1 (most synthetically accessible). Across varying methods with different reward functions, reaction-based methods generated compounds that are more synthetically accessible than atom/bond-based methods (MolDQN, nc3D-MolDQN, and c3D-MolDQN).

It is also interesting to note that for the 2D-DQN methods, the gap between MolDQN and RxnMolDQN versus MolDQN and RxnDQN is ~3.4. Although in the case of 3D-DQN methods, this gap ranges only between ~0.2 to 0.5. This suggests that when using QED (Section 2.2.11) as the reward function, there is a great favorability for the hybrid action space to perform reaction-based actions over atom/bond-based actions. However, this favorability barely exists for structure-based methods with LE or nSuCOS as the reward. This likely implies that the 3D-based rewards are balancing out the atom/bond-based and reaction-based actions during exploration of chemical space.

For 2D-DQN methods (Table 5.2), all results for synthetic accessibility including baselines using QED and penalized logP as a reward function are shown in Appendix E, Figures E1-E4.

6.4 Reward (QED)

Table 6.1 shows the results for 2D-DQN methods when optimizing for druglikeness using QED (Section 2.2.11) as the reward (Section 5.5.3). The top three best QED values are almost identical for 2D-DQN methods (see Best QED, 2nd Best QED, and 3rd Best QED in Table 6.1). However, when considering the average of these QED values across 100 DrugBank drugs (Figure 5.1), the values improved from MolDQN to RxnDQN to RxnMolDQN (see Average of Best QED, Average of 2nd Best QED, and Average of 3rd Best QED in Table 6.1). The difference in QED of compounds generated is more evident when considering the overall average QED. When considering all compounds designed by each method, it becomes clear that the reactions-based methods produce compounds with better average QED reward values than MolDQN: RxnMolDQN (0.671) > RxnDQN (0.659) > MolDQN (0.418) (see Overall Average QED in Table 6.1). All results for reward including baselines using QED as a reward function are shown in

Appendix F, Table F1. The top 10 scoring QED compounds for the selected 10 DrugBank drugs (Appendix B, Figure B3) are shown in Appendix F, Figures F2-F11.

Table 6.1: QED for 2D-DQN methods computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED. Note that these methods correspond to the "best methods" (Section 6.1). The values are given to three decimal places. Average of Best QED computes for the average of best QED per drug across 100 DrugBank compounds (similar for Average of 2nd Best QED and Average of 3rd Best QED). Overall average QED calculates for the average QED for all molecules generated across 100 DrugBank compounds (Appendix B, Figure B2). Performance in QED improved in the order of MolDQN to RxnDQN to RxnMolDQN.

Method	Best QED	Average of Best QED	2nd Best QED	Average of 2nd Best QED	3rd Best QED	Average of 3rd Best QED	Overall Average QED
MolDQN	0.947	0.849±0.125	0.947	0.844±0.126	0.947	0.841±0.126	0.418±0.275
RxnDQN	**0.948**	0.924±0.086	**0.948**	0.923±0.086	**0.948**	0.923±0.086	0.659±0.258
RxnMolDQN	**0.948**	**0.944±0.004**	**0.948**	**0.943±0.004**	**0.948**	**0.943±0.005**	**0.671±0.274**

6.5 Reward (Penalized logP)

Table 6.2 shows the results for 2D-DQN methods when optimizing for penalized logP (Section 2.2.10) as the reward (Section 5.5.3). The top three best penalized logP values are indistinguishable for 2D-DQN methods (see Best p.logP, 2nd Best p.logP, and 3rd Best p.logP in Table 6.2). When considering the average of these penalized logP values across 100 drugs from DrugBank (Figure 5.1), the values improved from RxnDQN to MolDQN to RxnMolDQN (see Average of Best p.logP, Average of 2nd Best p.logP, and Average of 3rd Best p.logP in Table 6.2). When considering all compounds designed by each method, my reactions-based methods produce compounds with better average reward values than MolDQN (RxnMolDQN: 6.207 > RxnDQN: 5.951 > MolDQN: 4.227; see Overall Average p.logP in Table 6.2). All results for reward including baselines using penalized logP as a reward function are shown in Appendix F, Table F12. The top 10 scoring penalized logP compounds for the selected 10 DrugBank drugs in Appendix B, Figure B3 are shown in Appendix F, Figures F13-F22.

Table 6.2: Penalized logP for 2D-DQN methods computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, penalized logP. Note that these methods correspond to the "best methods" (Section 6.1). The values are approximated to three decimal places. Average of Best p.logP computes for the average of best penalized logP per drug across 100 DrugBank compounds (similar for Average of 2nd Best p.logP and Average of 3rd Best p.logP). Overall average penalized logP calculates for the average penalized logP for all molecules generated across 100 DrugBank compounds (Appendix B, Figure B2). Performance in penalized logP improved in the order of MolDQN to RxnDQN to RxnMolDQN.

Method	Best p.logP	Average of Best p.logP	2nd Best p.logP	Average of 2nd Best p.logP	3rd Best p.logP	Average of 3rd Best p.logP	Overall Average p.logP
MolDQN	**10.000**	9.943±0.409	9.999	9.935±0.413	**9.999**	9.930±0.413	4.227±6.503
RxnDQN	**10.000**	9.420±2.528	9.999	9.415±2.527	**9.999**	9.412±2.526	5.951±4.768
RxnMolDQN	**10.000**	**9.995±0.005**	**10.000**	**9.990±0.007**	**9.999**	**9.986±0.009**	**6.207±4.747**

6.6 Reward (LE)

6.6.1 nc3D-DQN Methods

Table 6.3 shows the results for nc3D-DQN methods when optimizing for predicted ligand efficiency using LE (Section 3.2.5) as the reward (Section 5.5.3). The top three best predicted ligand efficiency values improved by more than 2.4-fold for reaction-based methods when comparing against atom/bond-based nc3D-MolDQN (see Best LE, 2nd Best LE, and 3rd Best LE in Table 6.3). Similar trends existed when considering the average of these values (see Average of Best LE, Average of 2nd Best LE, Average of 3rd Best LE in Table 6.3). When considering all compounds designed by each method, the hybrid-based nc3D-RxnMolDQN generated at least 2.1-fold better average reward values than nc3D-RxnDQN and nc3D-MolDQN (nc3D-RxnMolDQN: 0.749 > nc3D-RxnDQN: 0.352 > nc3D-MolDQN: 0.320; see Table 6.3). The difference in the "Overall Average LE" for nc3D-MolDQN and nc3D-RxnDQN is only 10%, but when considering "Average of Top 100 LE", nc3D-RxnDQN improved nc3D-MolDQN by approximately 1.5-fold (nc3D-RxnDQN: 0.578 > nc3D-MolDQN: 0.396; see Table 6.3). The difference between "Average of Top 100 LE" and "Overall Average LE" is greater for reaction-

based methods as compared to nc3D-MolDQN. This result is reasonable since reaction-based methods can make larger chemical modifications than atom/bond-based nc3D-MolDQN, which in effect can lead to larger standard deviation in predicted ligand efficiency. Overall, reaction-based nc3D-DQN methods are better at optimizing for predicted ligand efficiency than atom/bond-based nc3D-MolDQN.

Table 6.3: Predicted ligand efficiency (LE) for nc3D-DQN methods computed using 51 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) with the reward function, LE. The values are approximated to three significant figures and the units are in kcal $mol^{-1}HA^{-1}$. Average of Best LE computes for the average of best LE per ligand across 51 Malhotra smaller ligands (similar for Average of 2nd Best LE and Average of 3rd Best LE). Average of Top 100 LE computes for the average of Top 100 LE per ligand across 51 Malhotra smaller ligands (Appendix B, Figure B5). Overall average LE calculates for the average LE for all molecules generated across 51 Malhotra smaller ligands. Performance in LE improved in the order of nc3D-MolDQN, nc3D-RxnDQN, and nc3D-RxnMolDQN.

Method	Best LE	Average of Best LE	2nd Best LE	Average of 2nd Best LE	3rd Best LE	Average of 3rd Best LE	Average of Top 100 LE	Overall Average LE
nc3D-MolDQN	0.744	0.506±0.123	0.744	0.492±0.122	0.744	0.483±0.120	0.396±0.104	0.320±0.108
nc3D-RxnDQN	1.80	1.21±0.333	1.80	1.20±0.340	1.80	1.16±0.368	0.578±0.380	0.352±0.304
nc3D-RxnMolDQN	**1.89**	**1.42±0.207**	**1.89**	**1.40±0.217**	**1.89**	**1.39±0.220**	**1.28±0.266**	**0.749±0.512**

By considering the top 10 scoring predicted ligand efficiency molecules such as the examples shown in Figure 6.2, it became clear that an additional filter on the number of heavy atoms is required in the future especially for reaction-based methods. These algorithms can generate very small molecules that lead to very high predicted ligand efficiency but are not druglike (*e.g.*, H_2O and NH_3). This resulted in unexpectedly high predicted ligand efficiency scores seen in Table 6.3. The remaining results for the top 10 scoring predicted ligand efficiency for the other four selected Malhotra smaller ligands in Appendix B, Figure B6 are shown in Appendix F, Figures F23-F26.

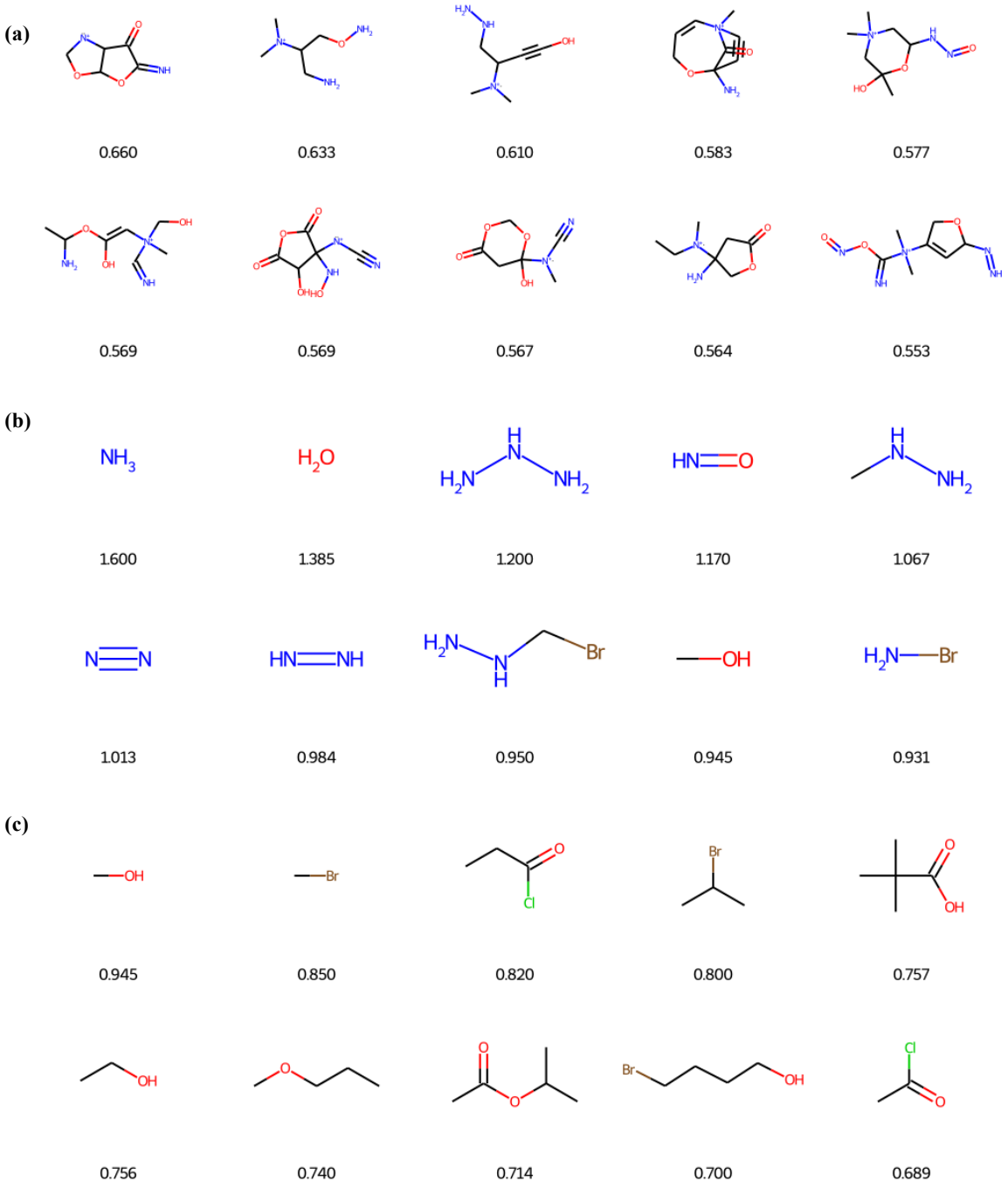


Figure 6.2: Top 10 predicted ligand efficiency scored compounds (in 2D) when using ACH/2ha4 from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is predicted ligand efficiency: **(a)** nc3D-MolDQN, **(b)** nc3D-RxnDQN, and **(c)** nc3D-RxnMolDQN. The units are in kcal $mol^{-1}HA^{-1}$. Reaction-based methods generated molecules in the range of two to seven heavy atoms.

6.6.2 c3D-DQN Methods

Table 6.4 shows the results for optimizing LE (Section 3.2.5) when using c3D-DQN methods (Section 4.3.1) against M^{pro}. The hybrid c3D-RxnMolDQN consistently performed the best except for "2nd Best LE" and "Overall Average LE" in Table 6.4. When comparing c3D-MolDQN and c3D-RxnDQN, it was found that the reaction-based c3D-RxnDQN performed at least 10% better for the top three best predicted ligand efficiency values (see Best LE, 2nd Best LE, and 3rd Best LE in Table 6.4). However, the average of these values followed the opposite trend (see Average of Best LE, Average of 2nd Best LE, Average of 3rd Best LE in Table 6.4). When considering all compounds designed by each method notwithstanding the large standard deviations, c3D-MolDQN generated at least 2.3-fold better average LE values than the reaction-based methods, c3D-RxnDQN and c3D-RxnMolDQN (c3D-MolDQN: 0.140±0.201 > c3D-RxnMolDQN: 0.061±0.329 > c3D-RxnDQN: -0.048±0.391; see Table 6.4). The reason for the negative average LE for c3D-RxnDQN is likely due to the binding pocket being a narrow channel, and the reactions make changes that are too big for the pocket.

Table 6.4: Predicted ligand efficiency (LE) for c3D-DQN methods computed using 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) with the reward function, LE. The values are approximated to three decimal places and the units are in kcal $mol^{-1}HA^{-1}$. Average of Best LE reports the average of the best LE per ligand across 17 covalent ligands (Appendix B, Figure B8); similarly, for Average of 2nd Best LE and Average of 3rd Best LE. Average of Top 100 LE computes the average of the Top 100 LE per ligand across the same 17 covalent ligands. Overall average LE refers to the average LE for all molecules generated across all 17 covalent ligands test cases (Appendix B, Figure B8). Performance in LE was best for c3D-RxnMolDQN, followed by c3D-MolDQN, and c3D-RxnDQN was worst.

Method	Best LE	Average of Best LE	2nd Best LE	Average of 2nd Best LE	3rd Best LE	Average of 3rd Best LE	Average of Top 100 LE	Overall Average LE
c3D-MolDQN	0.335	0.270±0.071	0.306	0.264±0.068	0.302	0.261±0.067	0.216±0.063	**0.140±0.201**
c3D-RxnDQN	0.385	0.265±0.090	**0.385**	0.256±0.085	0.330	0.250±0.080	0.157±0.116	-0.048±0.391
c3D-RxnMolDQN	**0.395**	**0.297±0.080**	0.339	**0.286±0.076**	**0.337**	**0.282±0.075**	**0.227±0.067**	0.061±0.329

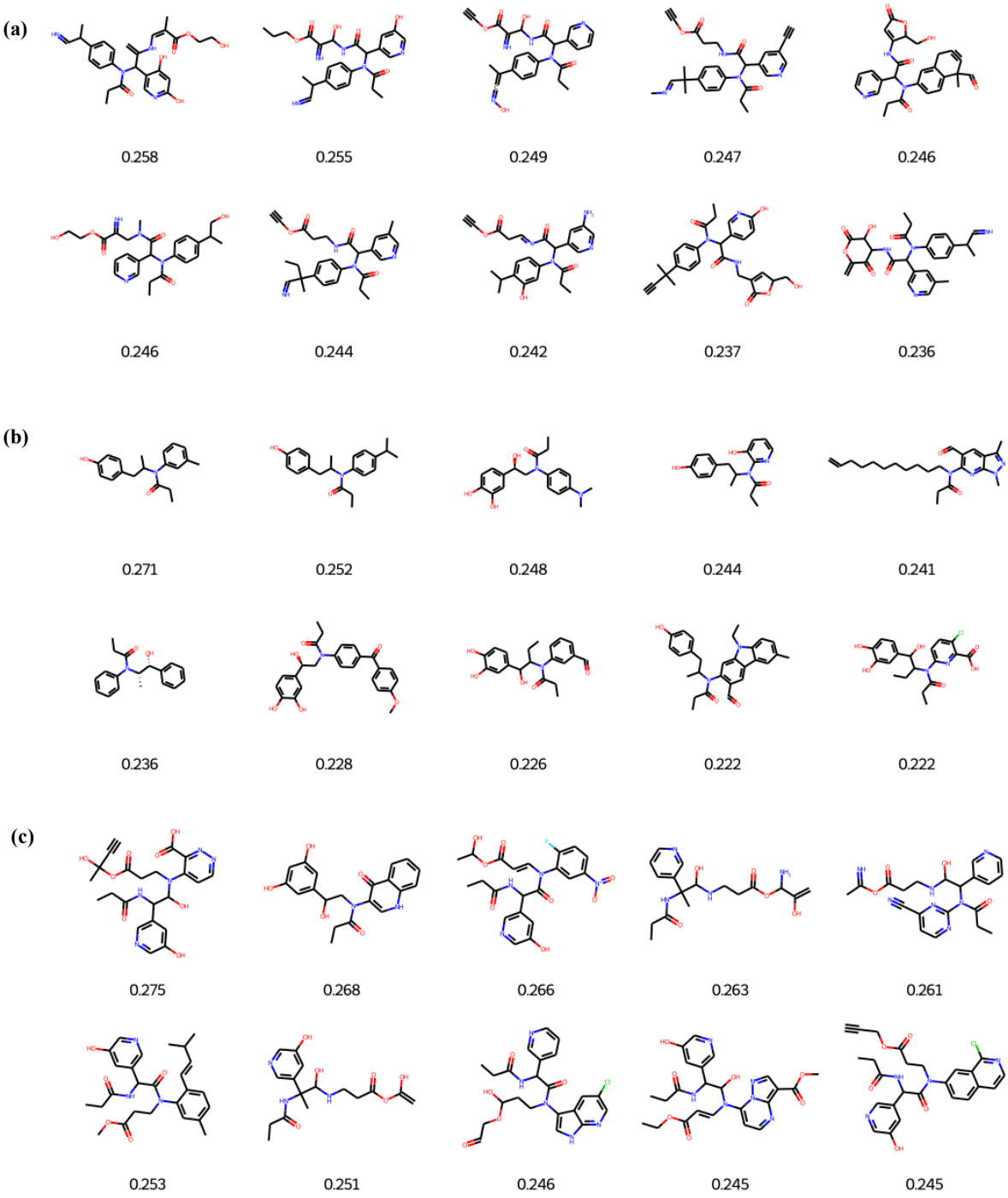


Figure 6.3: Top 10 predicted ligand efficiency scored compounds when using Mpro-x3110_0A from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) as a starting molecule where the reward function is predicted ligand efficiency: **(a)** c3D-MolDQN, **(b)** c3D-RxnDQN, and **(c)** c3D-RxnMolDQN. The units are in kcal $mol^{-1}HA^{-1}$.

Given the high standard deviation reported in the overall average LE for these methods, it is more informative to consider the average of top 100 molecules generated per ligand for each method (see Average of Top 100 LE in Table 6.4). In this scenario, c3D-RxnMolDQN (0.227) generated 40% and 10% better average reward values than c3D-RxnDQN (0.157) and c3D-MolDQN (0.216), respectively. These results suggest that for covalent ligand design in SARS-CoV-2 M^{pro}, it is harder especially for the pure reaction-based method (*i.e.*, c3D-RxnDQN) to optimize for predicted ligand efficiency. This is likely due to the restrictions placed upon the possible chemical modifications required to keep the covalent warhead bound to Cys-145. Overall, the hybrid c3D-RxnMolDQN method is better at optimizing for predicted ligand efficiency than just atom/bond-based c3D-MolDQN or reaction-based c3D-RxnDQN actions alone.

In the case of c3D-DQN methods, the constrained nature of these algorithms enables generation of compounds that are less likely to deviate from the original starting molecule as compared to nc3D-DQN methods (Section 6.6.1) regardless of the type of action spaces been used. This is exemplified from the examples of top 10 scoring predicted ligand efficiency compounds generated for c3D-DQN methods (Figure 6.3). The remaining top 10 scoring predicted ligand efficiency for the other selected two covalent ligands in Appendix B, Figure B9 are shown in Appendix F, Figures F27-28.

6.7 Reward (Normalized SuCOS)

Table 6.5 shows the results for nc3D-DQN methods when optimizing for normalized SuCOS (nSuCOS, Section 3.2.7) as the reward (Section 5.5.3). The top three best values are scored by the reaction-based methods (see Best nSuCOS, 2nd Best nSuCOS, and 3rd Best nSuCOS in Table 6.5). However, when considering the average of these values as well as the "Overall Average

nSuCOS", nc3D-MolDQN performed better than the reaction-based methods. These results suggest that on average, it is easier to retain the shape and feature overlaps (*i.e.*, pharmacophoric-based features) with the atom/bond-based action space as compared to the reaction-based action space where the reaction-based methods are more prone to make larger chemical modifications. Based on the overall average nSuCOS, since the values are lower for the reaction-based methods, it suggests that in the hybridized action space, it tends to favor the reaction-based actions than the atom/bond-based actions. Despite the difference in the action spaces, the overall average nSuCOS only differs by a maximum of 10%, which implies that the performance in single-objective optimization using nSuCOS is similar across all nc3D-DQN (nSuCOS) methods. The top 10 scoring normalized SuCOS for the selected five Malhotra smaller ligands in Appendix B, Figure B11 are shown in Appendix F, Figures F29-F33.

Table 6.5: Normalized SuCOS (nSuCOS) for nc3D-DQN methods computed using 45 Malhotra smaller ligands (Malhotra & Karanicolas, 2017) with the reward function, nSuCOS. The values are given to three significant figures. Average of Best nSuCOS computes for the average of best nSuCOS per ligand across 45 Malhotra smaller ligands (similarly for Average of 2nd Best nSuCOS and Average of 3rd Best nSuCOS). Overall average nSuCOS calculates the average nSuCOS for all molecules generated across 45 Malhotra smaller ligands (Appendix B, Figure B10). Performance in normalized SuCOS improved in the order of nc3D-RxnMolDQN, nc3D-RxnDQN, and nc3D-MolDQN.

Method	Best nSuCOS	Average of Best nSuCOS	2nd Best nSuCOS	Average of 2nd Best nSuCOS	3rd Best nSuCOS	Average of 3rd Best nSuCOS	Overall Average nSuCOS
nc3D-MolDQN	0.777	**0.549±0.149**	0.761	**0.533±0.145**	0.759	**0.524±0.144**	**0.286±0.159**
nc3D-RxnDQN	0.842	0.497±0.154	**0.791**	0.473±0.150	**0.786**	0.461±0.151	0.259±0.137
nc3D-RxnMolDQN	**0.882**	0.509±0.146	**0.791**	0.481±0.138	**0.786**	0.472±0.140	0.254±0.136

6.8 Walters' Compound Quality

Table 6.6 shows the results for average Walters' compound quality (Section 5.5.4) for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods with varying reward functions and

docking types (if relevant): 2D-DQN (QED), nc3D-DQN (LE), nc3D-DQN (nSuCOS), and c3D-DQN (LE) methods. The average of Walters' compound quality for both RxnDQN-based and RxnMolDQN-based methods improved by at least 2.2-fold as compared to MolDQN-based methods regardless of the types of reward function and docking. When considering all compounds designed by each method, my reactions-based methods produced compounds with better average Walters' compound quality than Google's MolDQN-based methods. Overall, 2D-DQN methods led to a higher average Walters' compound quality than the corresponding action spaces in 3D-DQN methods. This is likely because the reward function of QED (Section 2.2.11) considers drug-likeness explicitly, which is one of the components of Walters' compound quality.

Table 6.6: Average Walters' compound quality for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods in 2D-DQN (QED), nc3D-DQN (LE), nc3D-DQN (nSuCOS), and c3D-DQN (LE). Note that for 2D-DQN (QED), these methods correspond to the "best methods" (Section 6.1). These values are computed using the dataset described in Figures 5.1 for 2D-DQN, 5.2 for nc3D-DQN, and 5.3 for c3D-DQN methods. The values are given to three decimal places. Average Walters' compound quality is computed by taking the average of Walters' compound quality for all molecules generated across the given dataset for starting molecules (Figures 5.1-5.3). Walters' compound quality improves from 0 (worst) to 1 (best). In general, RxnDQN-based and RxnMolDQN-based methods generated compounds with better Walters' compound quality than MolDQN-based methods (MolDQN, nc3D-MolDQN, and c3D-MolDQN).

Method (Reward)	Average Walters' Compound Quality for MolDQN-based Method (/1)	Average Walters' Compound Quality for RxnDQN-based Method (/1)	Average Walters' Compound Quality for RxnMolDQN-based Method (/1)
2D-DQN (QED)	0.183±0.097	**0.624±0.178**	0.600±0.110
nc3D-DQN (LE)	0.096±0.063	**0.584±0.227**	0.437±0.128
nc3D-DQN (nSuCOS)	0.092±0.081	**0.424±0.234**	0.239±0.093
c3D-DQN (LE)	0.140±0.112	**0.563±0.079**	0.308±0.077

For 2D-DQN methods, all results for Walters' compound quality including baselines using QED and penalized logP as a reward function are shown in Appendix G, Tables G1 and G2, respectively.

6.9 Rediscovery

6.9.1 2D-DQN Methods

Table 6.7 shows the results for rediscovery (Section 5.5.7) for 2D-DQN methods. MolDQN performed better in rediscovery than RxnDQN and RxnMolDQN. This is because atom/bond-based action space can either perform no modification or smaller chemical modifications (*i.e.*, one atom or bond at a time) to a molecule, so it is much easier to achieve rediscovery than the reaction-based action space.

Table 6.7: Rediscovery for 2D-DQN methods computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED. These methods correspond to the "best methods" (Section 6.1). MolDQN performs the best in rediscovery, but when considering other factors such as synthetic accessibility and chemical space coverage, RxnMolDQN is the best method.

Method	Success Rediscovery (/100)
MolDQN	**96**
RxnDQN	0
RxnMolDQN	3

Rediscovery of 0 for RxnDQN is not surprising since two conditions must be satisfied. Firstly, the fragment of the DrugBank molecule that got disconnected needs to be present inside the reagents set. Secondly, even if the fragment is in the reagents set, it needs to be selected from 50 randomly chosen reagents to perform the chemical modification for each reaction. Consequently, even if the first criterion is met, the chance of satisfying the second criterion simultaneously is extremely low. Therefore, the rediscovery of 3 for RxnMolDQN is likely achieved by the atom/bond-based action space. Hybrid RxnMolDQN strikes a balance for 2D-DQN methods since it incorporates the benefit of MolDQN to achieve higher rediscovery than RxnDQN and it promotes greater synthetic accessibility (Section 6.2) and chemical space coverage (Section 6.11)

than MolDQN. All results for rediscovery including the baselines for both QED and penalized logP as a reward function are shown in Appendix I, Tables I1 and I2, respectively.

6.9.2 nc3D-DQN (LE) Methods

Tables 6.8 and 6.9 show the results for rediscovery (Section 5.5.7) of Malhotra smaller and larger ligands for nc3D-DQN (LE) methods, respectively. Reaction-based nc3D-RxnDQN and nc3D-RxnMolDQN can rediscover the starting molecule (*i.e.*, Malhotra smaller ligand) unlike nc3D-MolDQN (Table 6.8). In fact, nc3D-RxnDQN is better at rediscovering the starting molecule than nc3D-RxnMolDQN. This suggests that the reaction-based action space is creating the capability for nc3D-DQN (LE) methods to rediscover the original starting molecule.

Table 6.8: Rediscovery of Malhotra smaller ligands for nc3D-DQN methods computed using 51 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) with the reward function, LE. nc3D-RxnDQN performs the best in rediscovery, followed by nc3D-RxnMolDQN and nc3D-MolDQN.

Method	Success Rediscovery (/51)
nc3D-MolDQN	0
nc3D-RxnDQN	**50**
nc3D-RxnMolDQN	6

Table 6.9: Rediscovery of Malhotra larger ligands for nc3D-DQN methods computed using 51 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) with the reward function, LE. nc3D-RxnDQN can grow from smaller to larger ligand despite of starting from smaller ligand.

Method	Success Rediscovery (/51)
nc3D-MolDQN	0
nc3D-RxnDQN	**3**
nc3D-RxnMolDQN	0

The results shown in Table 6.9 is a more challenging problem than in Table 6.8, since the starting molecule (*i.e.*, Malhotra smaller ligand) needs to grow and generate Malhotra larger ligand of the

same ligand-pair. Despite of this difficult task, nc3D-RxnDQN can successfully achieve this goal. This further suggests that nc3D-RxnDQN has the capability to make informed decision about the region of chemical space to explore and act as a potential alternative to fragment-based drug discovery (Section 3.2.6). It is not surprising that nc3D-MolDQN and nc3D-RxnMolDQN cannot achieve this type of rediscovery due to the frequency of occurrence for rediscovery already reported in Table 6.8 and no information about the target (*i.e.*, larger ligand) is given during the tests (Section 5.4.4).

6.9.3 nc3D-DQN (nSuCOS) Methods

Tables 6.10 and 6.11 show the results for rediscovery (Section 5.5.7) of Malhotra smaller and larger ligands, respectively, for nc3D-DQN (nSuCOS) methods. Unlike nc3D-DQN (LE) methods, all methods in nc3D-DQN (nSuCOS) methods can achieve rediscovery for smaller ligands.

Table 6.10: Rediscovery of Malhotra smaller ligands for nc3D-DQN methods computed using 45 Malhotra smaller ligands (Malhotra & Karanicolas, 2017) with the reward function, normalized SuCOS. nc3D-RxnDQN performs the best in rediscovery, followed by nc3D-RxnMolDQN and nc3D-MolDQN.

Method	Success Rediscovery (/45)
nc3D-MolDQN	2
nc3D-RxnDQN	**43**
nc3D-RxnMolDQN	25

Table 6.11: Rediscovery of Malhotra larger ligands for nc3D-DQN methods computed using 45 Malhotra smaller ligands (Malhotra & Karanicolas, 2017) with the reward function, normalized SuCOS. nc3D-DQN methods cannot grow from smaller to larger ligands by using normalized SuCOS as the reward.

Method	Success Rediscovery (/45)
nc3D-MolDQN	0
nc3D-RxnDQN	0
nc3D-RxnMolDQN	0

When comparing nc3D-DQN methods with LE or nSuCOS as the reward, the rate of rediscovery of Malhotra smaller ligands improved for nc3D-MolDQN and nc3D-RxnMolDQN when using the reward function of nSuCOS (Table 6.10) as compared to LE (Table 6.8). However, the rate of rediscovery of Malhotra smaller and larger ligands was higher for nc3D-RxnDQN (LE) as compared to nc3D-RxnDQN (nSuCOS). This suggests that the reward function of normalized SuCOS assisted in rediscovery for methods that involved atom/bond-based action space, but not for the pure reaction-based action space. In addition, neither of nc3D-DQN (nSuCOS) methods were successful in achieving rediscovery of Malhotra larger ligands (Table 6.11) unlike nc3D-RxnDQN (LE) in Table 6.9. This indicates that despite providing the information about both smaller and larger receptors when preparing receptor PDBQT files (Section 5.4.4), it did not lead to improvement in rediscovery of larger ligands. Therefore, it is difficult to achieve rediscovery with nc3D-DQN methods when using normalized SuCOS (Section 3.2.7) as the reward.

6.9.4 c3D-DQN Methods

Table 6.12 shows the results for rediscovery (Section 5.5.7) for c3D-DQN methods. All c3D-DQN methods were able to rediscover the original covalent ligands for SARS-CoV-2 main protease. This result may be assisted by the nature of the constrained covalent docking to maintain covalent warhead binding at the active site Cys-145 (Section 4.3.1). Therefore, the exploration of chemical space may be further guided to explore the region that is closer to the original starting molecule. Even though these methods can achieve rediscovery, reaction-based c3D-DQN methods performed better than c3D-MolDQN. This suggests that the reaction-based actions are assisting in the hybrid method to guide the exploration of chemical space towards the desired covalent ligand.

Table 6.12: Rediscovery for c3D-DQN methods computed using 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) with the reward function, LE. All c3D-DQN methods were able to achieve rediscovery, but reaction-based c3D-DQN methods performed better than atom/bond-based c3D-MolDQN.

Method	Success Rediscovery (/17)
c3D-MolDQN	6
c3D-RxnDQN	**16**
c3D-RxnMolDQN	12

6.10 Tanimoto Similarity

6.10.1 Representative Examples for Top 10 Tanimoto Similarity

For 2D-DQN methods, the representative examples of top 10 Tanimoto similarity compounds (Section 6.10.2) to the initial starting drug are shown using etravirine (Figure 6.4). Etravirine is a non-nucleoside reverse transcriptase inhibitor used for the treatment of HIV. This drug is one of the ten drugs selected from 100 DrugBank compounds (Figure 5.1) based on the drug's binding target (Appendix B, Figure B3).

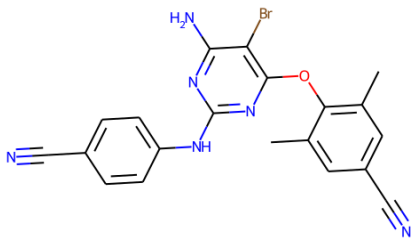


Figure 6.4: DrugBank drug selected to show representative examples of Tanimoto similarity between generated and initial starting drug molecules for 2D-DQN methods: Etravirine (DrugBank ID: DB06414).

For nc3D-DQN (LE) methods, the representative examples for top 10 Tanimoto similarity (Section 6.10.2) to the Malhotra smaller ligands are shown using the receptor of human carbonic anhydrase II (hCA II) (PDB ID: 2wej, Figure 6.5).

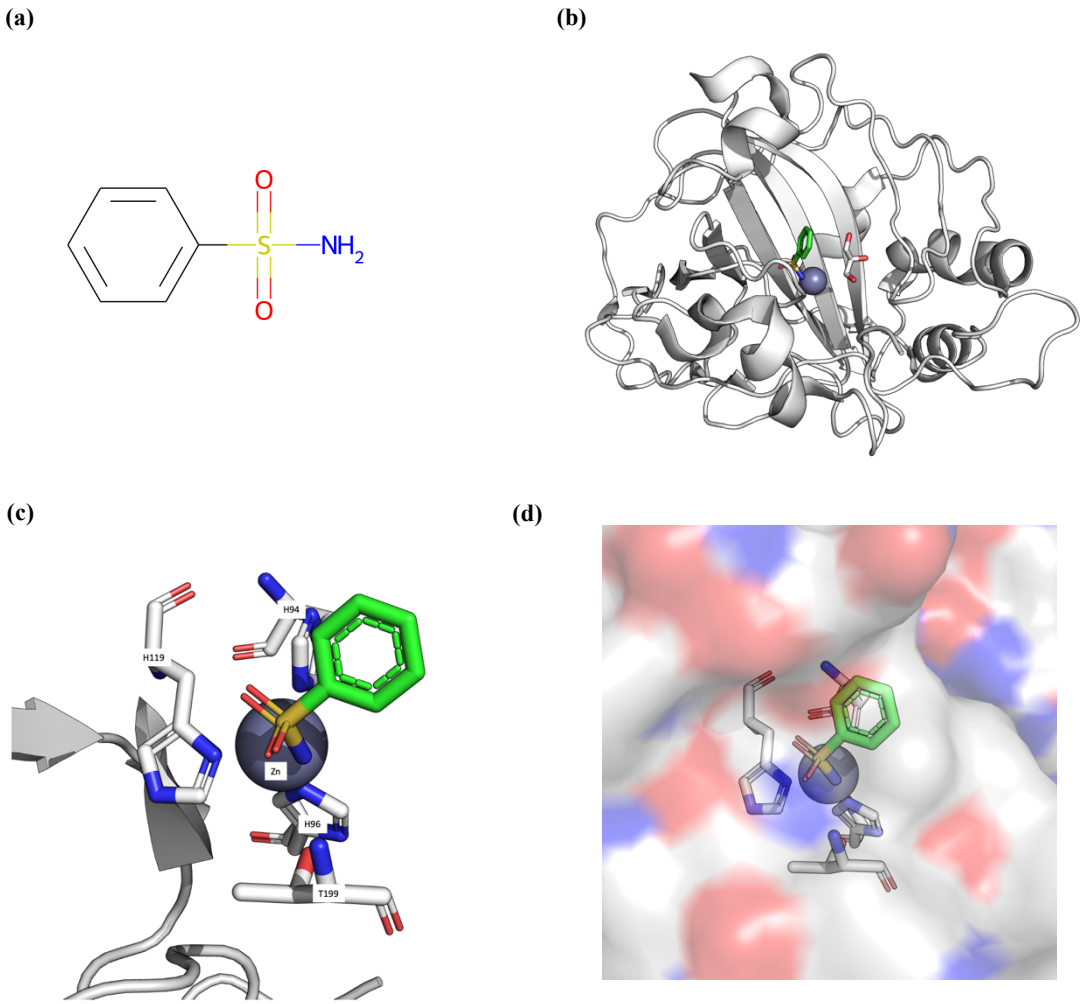


Figure 6.5: (a) Malhotra smaller ligand selected to show representative top 10 Tanimoto similarity compounds for nc3D-DQN (LE) methods: FB2; **(b)** The crystal structure of human carbonic anhydrase II (hCA II) (PDB ID: 2wej); **(c)** Zoom in of binding site showing H94, H96, H119, and T199 (white) and bound FB2 (green) (Scott *et al.*, 2009) with the catalytic Zn^{2+} ion (blue purple); **(d)** The molecular surface with bound FB2 (green) and the binding site. Figures **(b)-(d)** were created using PyMOL (Schrödinger, 2022).

hCA II is a zinc-containing metalloenzyme that catalyzes the interconversion between carbon dioxide and water and the dissociated ions of carbonic acid (*i.e.*, hydrogen ions and bicarbonate) (Badger and Price, 1994). The ligand bound to hCA II (Figure 6.5(a)) is one of the five ligands from 51 Malhotra smaller ligands (Figure 5.2) selected based on the molecules' chemical diversity (Appendix B, Figure B6). Figure 6.5(b) shows the whole protein of hCA II (*i.e.*, monomer (grey) and asymmetric symmetry) together with its binding site (white) in Figure 6.5(c). Figure 6.5(d)

shows the molecular surface of the binding pocket in hCA II and the known active ligand of FB2 (*i.e.*, Malhotra smaller ligand) bound.

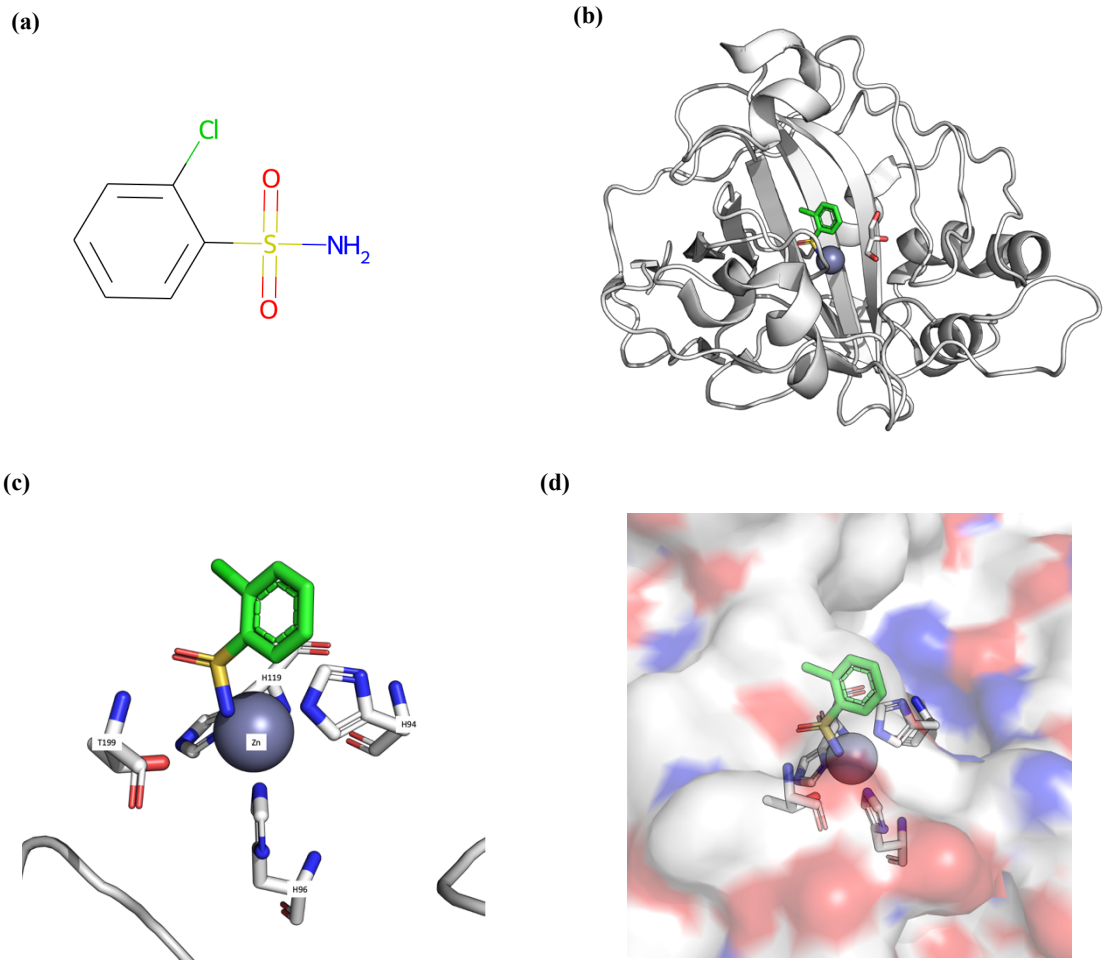


Figure 6.6: (a) Malhotra smaller ligand selected to show representative top 10 Tanimoto similarity compounds for nc3D-DQN (nSuCOS) methods: FB1; **(b)** The crystal structure of human carbonic anhydrase II (hCA II) (PDB ID: 2weh); **(c)** Zoom in of binding site showing H94, H96, H119, and T199 (white) and bound FB1 (green) (Scott *et al.*, 2009) with the catalytic Zn^{2+} ion (blue purple); **(d)** The molecular surface with bound FB1 (green) and the binding site. Figures **(b)-(d)** were created using PyMOL (Schrödinger, 2022).

For nc3D-DQN (nSuCOS) methods, the representative top 10 results for Tanimoto similarity are shown for the same target as nc3D-DQN (LE) methods (Figure 6.5). However, this time using the

PDB ID of 2weh instead of 2wej (Figure 6.6). The bound ligand is the known active ligand of FB1 (*i.e.*, Malhotra smaller ligand), which is one of the five ligands selected from 45 Malhotra smaller ligands based on molecules' structural diversity (Appendix B, Figure B11). Figure 6.6 shows a similar type of figures as Figure 6.5.

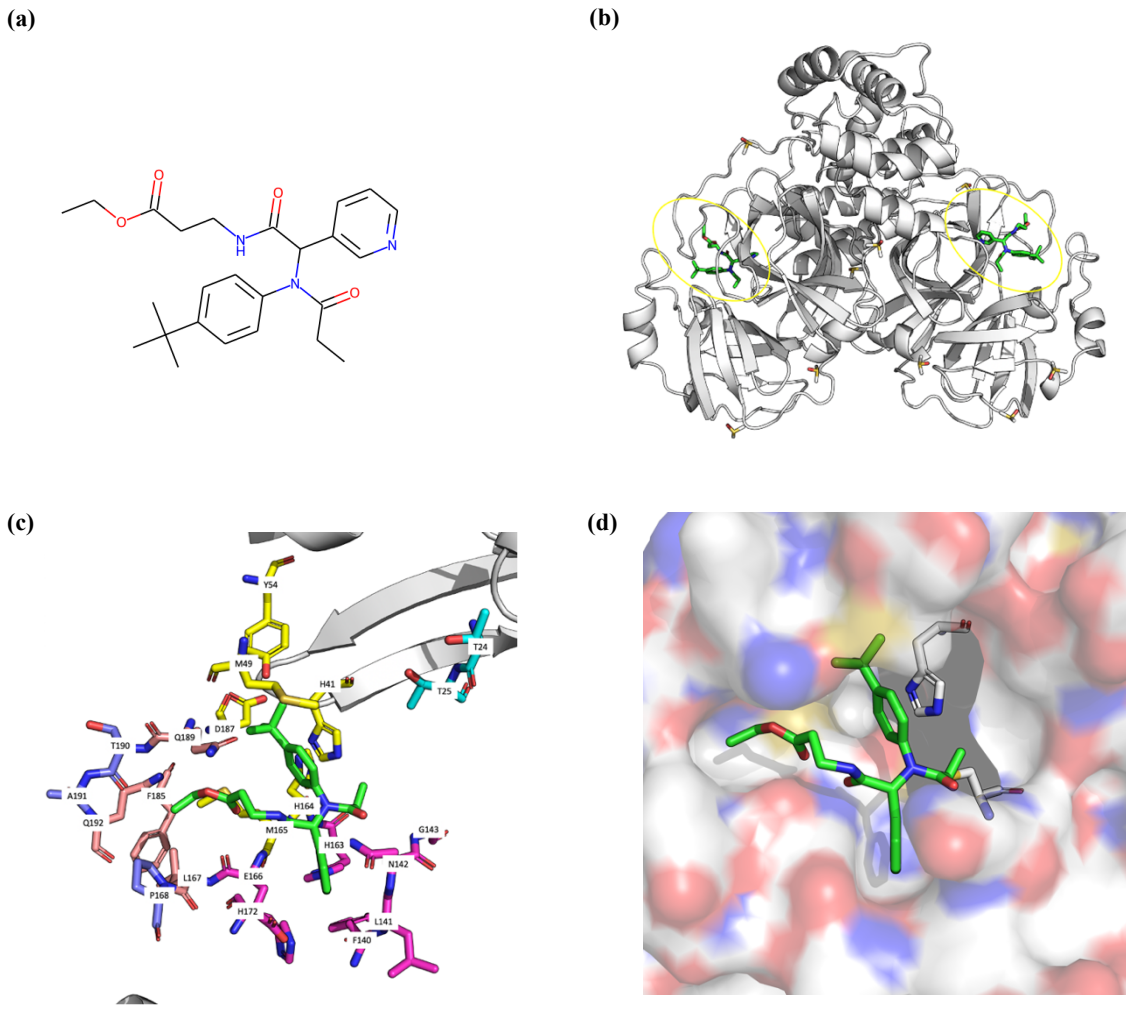


Figure 6.7: (a) Covalent ligand selected to show representative top 10 Tanimoto similarity compounds for c3D-DQN methods: Mpro-x3110_0A; **(b)** The crystal structure of SARS-CoV-2 main protease (Douangamath *et al.*, 2020) complexed with Mpro-x3110_0A showing the whole dimeric protein and its secondary structure where the binding site is circled (yellow); **(c)** close up of the binding side chains site, showing S1′ side chain (cyan; T24, T25), S1 side chain (magenta; F140, L141, N142, G143, E166, H163, and H172), S2 side chain (yellow; H41, M49, Y54, H164, M165, and D187), S4 sidechain (wheat; L167, F185, Q189, and Q192), S5 side chain (blue-purple; P168, T190, and A191), and the covalent ligand Mpro-x3110_0A (green); **(d)** The molecular surface with Mpro_x3110_0A (green) and the catalytic dyad residues (His-41 and Cys-145). Figures **(b)-(d)** were created using PyMOL (Schrödinger, 2022).

For c3D-DQN methods, the representative top 10 Tanimoto similarity results are shown using the crystal structure of SARS-CoV-2 M^{pro} complexed with Mpro-x3110_0A (Figure 6.7(b)) together with its binding site (Figure 6.7(c)) and molecular surface (Figure 6.7(d)). This ligand (Figure 6.7(a)) is one of the three covalent ligands selected from 17 covalent ligands of SARS-CoV-2 M^{pro} based on molecules' structural diversity (Appendix B, Figure B9).

6.10.2 Tanimoto Similarity Results for DQN Methods

Table 6.13 shows the results for average Tanimoto similarity (Section 5.5.11) between the generated molecule and the starting DrugBank compounds (Wishart *et al.*, 2018) for the 2D-DQN methods. Reaction-based methods improved these values by more than 2.1-fold as compared to MolDQN. Therefore, while reaction-based methods make larger chemical modifications than the atom/bond-based method, both RxnDQN and RxnMolDQN are better at generating compounds that are more structurally similar to the initial drug than MolDQN.

Table 6.13: Average Tanimoto similarity for 2D-DQN methods computed between initial 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) and the generated molecule at each episode using the reward function, QED. These methods correspond to the "best methods" (Section 6.1). The values are given to three decimal places. Tanimoto similarity ranges from 0 (least structurally similar) to 1 (most structurally similar). RxnDQN and RxnMolDQN generated compounds that are more structurally similar than MolDQN.

Method	Top 1 Average Tanimoto Similarity	Top 10 Average Tanimoto Similarity	Top 20 Average Tanimoto Similarity
MolDQN	0.251±0.159	0.206±0.115	0.195±0.107
RxnDQN	**0.585±0.325**	**0.529±0.359**	**0.513±0.370**
RxnMolDQN	0.573±0.321	0.460±0.295	0.405±0.260

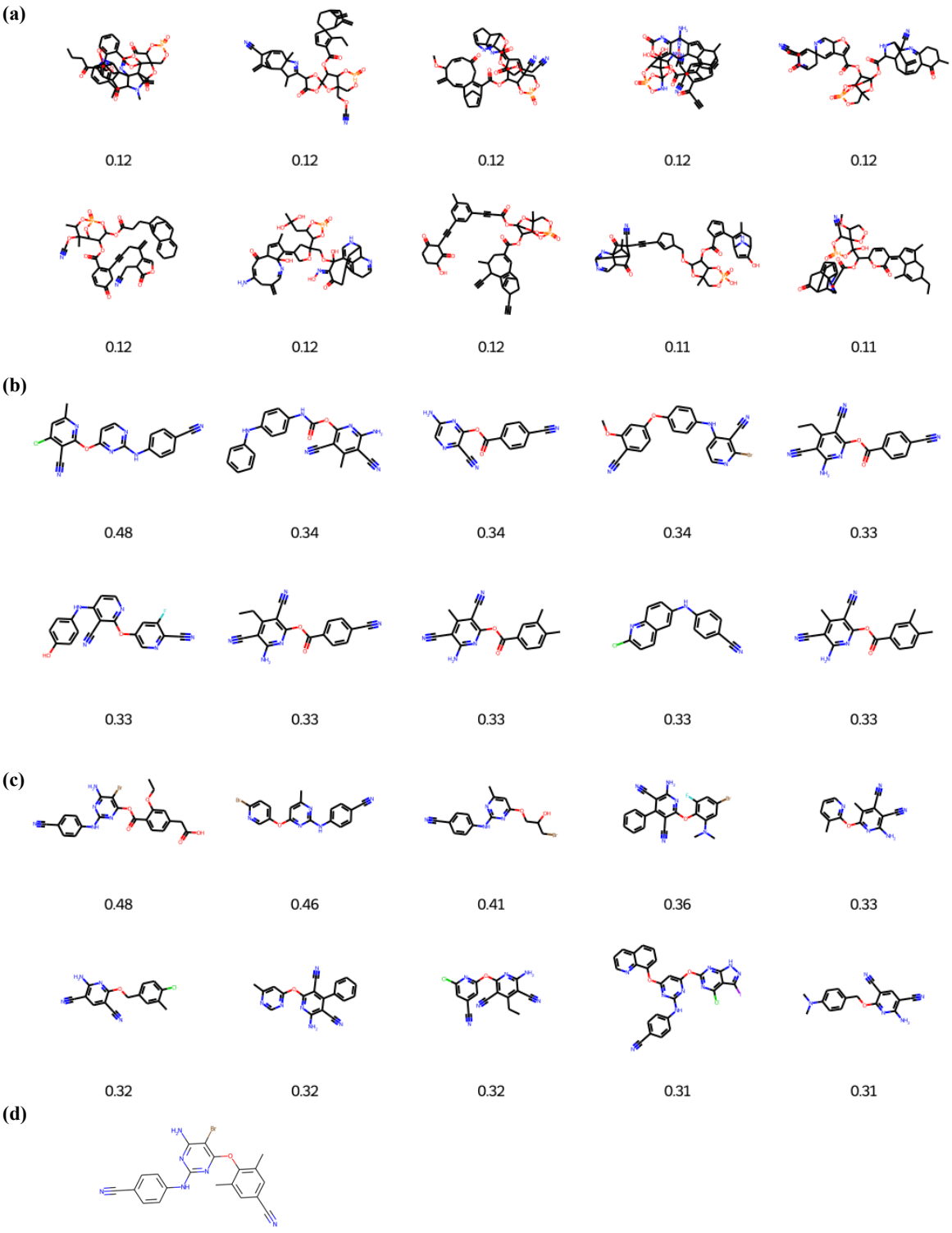


Figure 6.8: Top 10 Tanimoto similarity compounds for 2D-DQN methods computed between etravirine/DB06414 from DrugBank **(d)** and generated molecule at each episode: **(a)** MolDQN, **(b)** RxnDQN, and **(c)** RxnMolDQN. MolDQN generated compounds that are 3-folds more distant to the initial starting drug than reaction-based methods. Compounds generated by MolDQN are also chemically unreasonable.

In addition, reaction-based methods can propose more chemically reasonable molecules as shown through the representative examples for top 10 Tanimoto similarity (Section 6.10.1) in Figure 6.8. This aligns with the observations made in Figure 6.1(a) for synthetic accessibility (*i.e.*, reaction-based methods generated more synthetically accessible compounds than MolDQN). The remaining results for top 10 Tanimoto similarity compounds from the selected drugs (Appendix B, Figure B3) are given in Appendix M, Figures M3-M11. All results for Tanimoto similarity to the DrugBank compounds including baselines using QED and penalized logP as a reward function are shown in Appendix M, Tables M1 and M2, respectively.

Table 6.14: Average Tanimoto similarity for nc3D-DQN (LE) methods computed between 51 Malhotra smaller or larger ligands (Malhotra and Karanicolas, 2017) and generated molecule at each episode using the reward function, LE. The larger ligand is from the same ligand-pair as the smaller ligand. The values are approximated to three decimal places. Tanimoto similarity improves from 0 (least structurally similar) to 1 (most structurally similar). nc3D-RxnDQN generated compounds that are more structurally similar than nc3D-MolDQN and nc3D-RxnMolDQN.

Method	**Smaller Ligands**			**Larger Ligands**		
	Top 1 Average Tanimoto Similarity	**Top 10 Average Tanimoto Similarity**	**Top 20 Average Tanimoto Similarity**	**Top 1 Average Tanimoto Similarity**	**Top 10 Average Tanimoto Similarity**	**Top 20 Average Tanimoto Similarity**
nc3D-MolDQN	0.509±0.118	0.397±0.109	0.359±0.104	0.348±0.089	0.292±0.076	0.272±0.072
nc3D-RxnDQN	**0.984±0.112**	**0.984±0.116**	**0.977±0.130**	**0.481±0.189**	**0.440±0.157**	**0.435±0.155**
nc3D-RxnMolDQN	0.545±0.225	0.338±0.165	0.281±0.142	0.365±0.135	0.273±0.095	0.242±0.084

In the case of nc3D-DQN (LE) methods, the average Tanimoto similarity (Section 5.5.11) results are computed between the generated molecule and Malhotra smaller or larger ligand as shown in Table 6.14. In both cases, nc3D-RxnDQN generated more structurally similar molecules to the corresponding ligand as compared to nc3D-RxnMolDQN and nc3D-MolDQN.

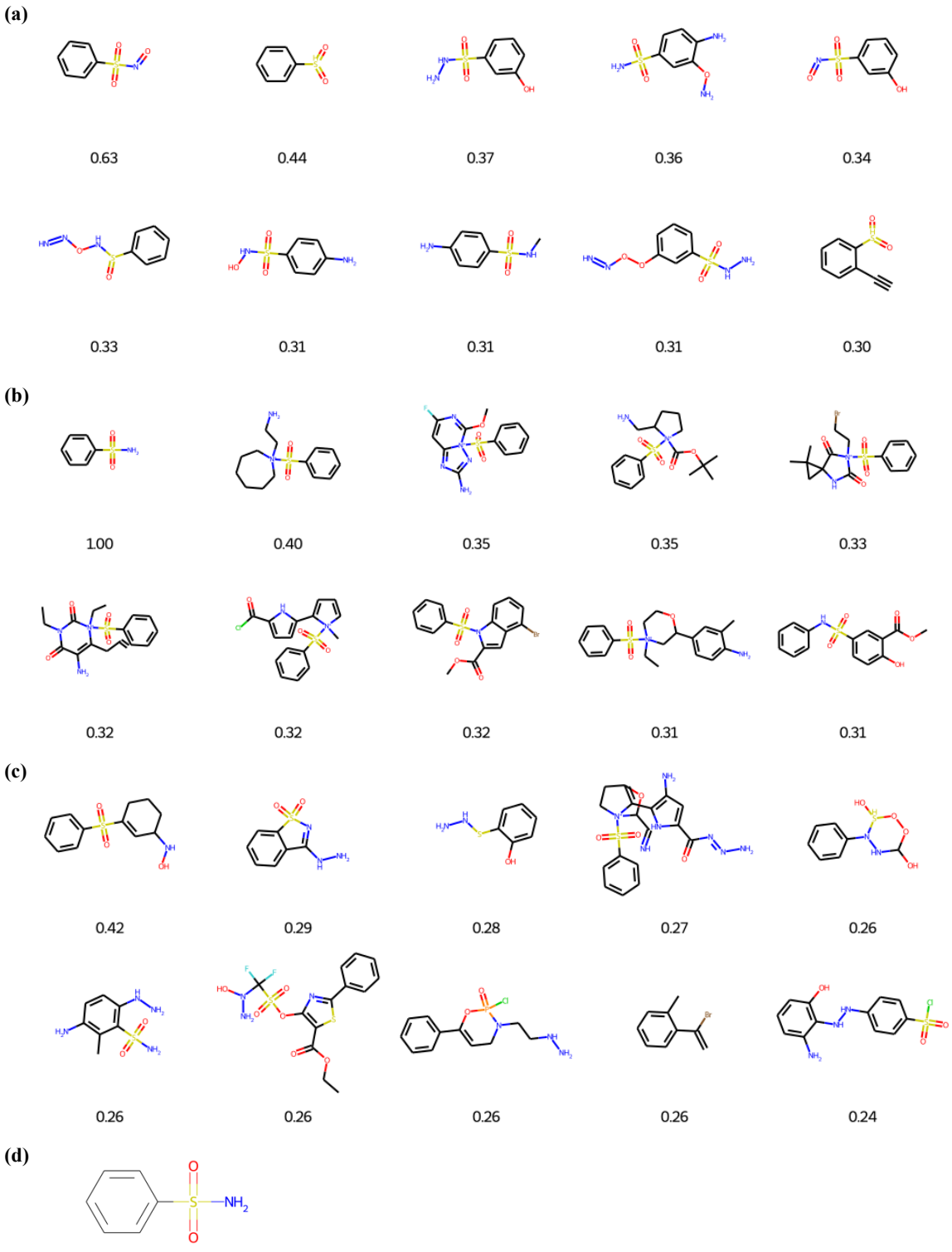


Figure 6.9: Top 10 Tanimoto similarity compounds for nc3D-DQN (LE) methods computed between FB2/2wej (Section 6.10.1) from the Malhotra set **(d)** and generated molecule at each episode: **(a)** nc3D-MolDQN, **(b)** nc3D-RxnDQN, and **(c)** nc3D-RxnMolDQN. The performance in Tanimoto similarity improved in the order of nc3D-RxnMolDQN, nc3D-MolDQN, and nc3D-RxnDQN.

When comparing the values between smaller and larger ligands, it was found that the average Tanimoto similarity was higher for the smaller ligands. This states that nc3D-DQN (LE) methods tend to explore a chemical space which is closer to the initial starting molecule as compared to the larger ligand (*i.e.*, target molecule). Unlike MolDQN (Figure 6.8), nc3D-MolDQN (LE) propose compounds that are much drug-like as shown through the representative example compounds for top 10 Tanimoto similarity (Section 6.10.1) in Figure 6.9. Therefore, there is an improvement in the quality of generated molecules when using the 3D-based reward function (LE) as compared to 2D-based reward function (QED) especially for MolDQN-based method. This observation also aligns with approximately 2-fold increase in average Tanimoto similarity values (*i.e.*, Malhotra smaller ligands in Tables 6.13-6.14) and synthetic accessibility reported in Figures 6.1(a)-(b) (blue box plots). The remaining results for the top 10 Tanimoto similarity compounds from the selected Malhotra smaller ligands (Appendix B, Figure B6) are included in Appendix M, Figures M12-M15. The representative 3D docked pose from AutoDock Vina for nc3D-DQN (LE) methods is shown in Figure 6.10 using one of the generated molecules from nc3D-RxnDQN.

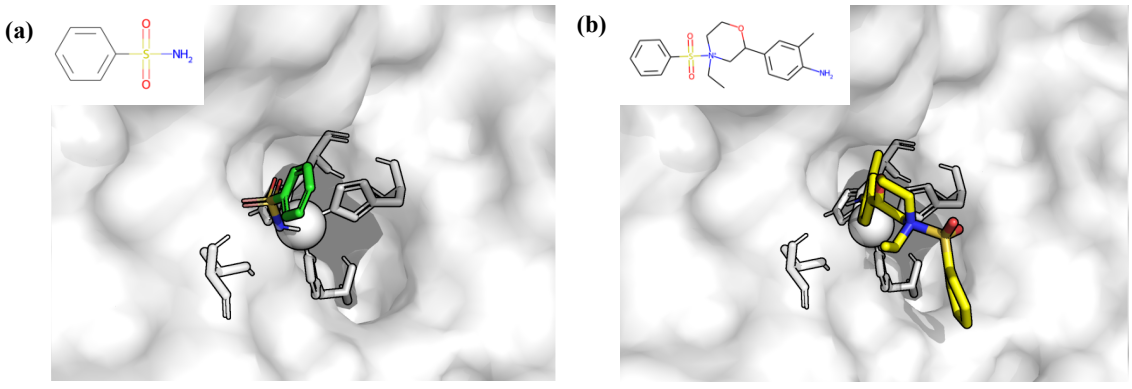


Figure 6.10: Representative 3D docked pose **(b)** for nc3D-DQN (LE) using the generated molecule (yellow) in nc3D-MolDQN with the initial starting molecule of FB2 (PDB ID: 2wej) **(a)** shown in green. H94, H96, H119, T199, and the catalytic Zn^{2+}ion are shown in white (Section 6.10.1).

For nc3D-DQN (nSuCOS) methods, the average Tanimoto similarity values (Table 6.15) are computed in a similar way as nc3D-DQN (LE) methods (Table 6.14), but using 45 smaller ligands instead of 51. For both smaller and larger ligands, nc3D-RxnDQN generated compounds that are structurally similar to the corresponding ligands than nc3D-MolDQN and nc3D-RxnMolDQN, which is a similar observation as nc3D-DQN (LE) methods (Table 6.14). When comparing these results against nc3D-DQN (LE) methods, the reward function of normalized SuCOS tends to generate compounds that are more structurally similar to both smaller and larger ligands. Similar type of chemical space exploration between smaller and larger ligands is likely to be performed for nc3D-DQN (nSuCOS) methods as described earlier for nc3D-DQN (LE) methods. The representative results for top 10 Tanimoto similarity are shown in Figure 6.11 and 3D docked pose from AutoDock Vina in Figure 6.12.

Table 6.15: Average Tanimoto similarity for nc3D-DQN (nSuCOS) methods computed between 45 Malhotra smaller or larger ligands (Malhotra and Karanicolas, 2017) and generated molecule at each episode using the reward function, nSuCOS. The larger ligand is from the same ligand-pair as the smaller ligand. The values are approximated to three decimal places. Tanimoto similarity improves from 0 (least structurally similar) to 1 (most structurally similar). nc3D-RxnDQN generated compounds that are more structurally similar than nc3D-MolDQN and nc3D-RxnMolDQN.

Method	Smaller Ligands			Larger Ligands		
	Top 1 Average Tanimoto Similarity	Top 10 Average Tanimoto Similarity	Top 20 Average Tanimoto Similarity	Top 1 Average Tanimoto Similarity	Top 10 Average Tanimoto Similarity	Top 20 Average Tanimoto Similarity
nc3D-MolDQN	0.586±0.136	0.467±0.122	0.426±0.114	0.364±0.098	0.318±0.089	0.300±0.085
nc3D-RxnDQN	**0.979±0.100**	**0.927±0.194**	**0.848±0.260**	**0.446±0.145**	**0.410±0.140**	**0.390±0.139**
nc3D-RxnMolDQN	0.800±0.247	0.510±0.244	0.421±0.210	0.411±0.112	0.325±0.093	0.296±0.088

Like the nc3D-DQN (LE) methods (Figure 6.9), the reward function of normalized SuCOS is also successful at generating more drug-like compounds for nc3D-MolDQN as compared to MolDQN (Figure 6.8). The remaining results for top 10 Tanimoto similarity of the selected ligands (Appendix B, Figure B6) are included in Appendix M, Figures M18-M21.

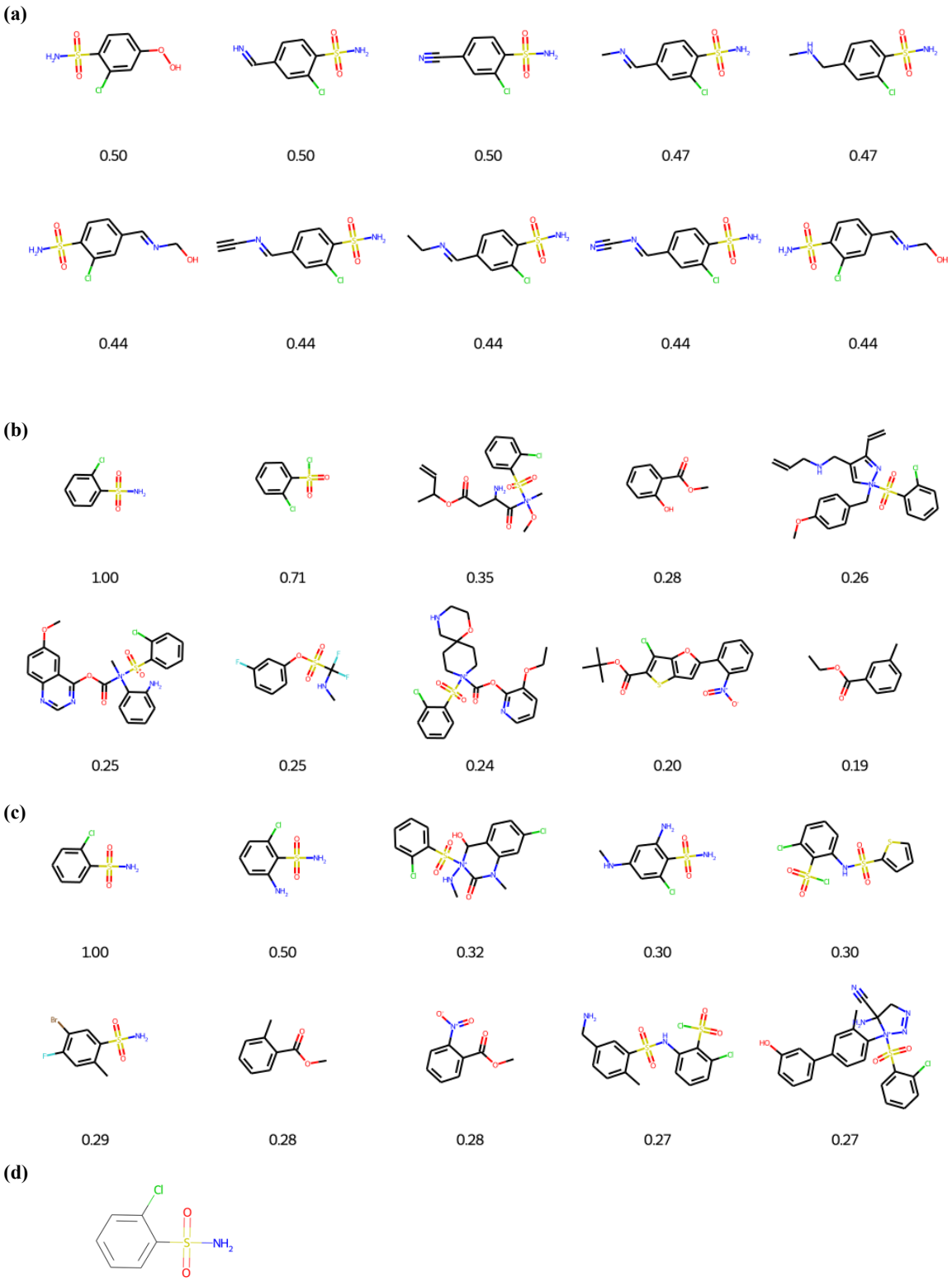


Figure 6.11: Top 10 Tanimoto similarity compounds for nc3D-DQN (nSuCOS) methods computed between FB1/2weh from the Malhotra set **(d)** and generated molecule at each episode: **(a)** nc3D-MolDQN, **(b)** nc3D-RxnDQN, and **(c)** nc3D-RxnMolDQN. Reaction-based methods performed better at generating molecules closer to the initial starting molecule than nc3D-MolDQN.

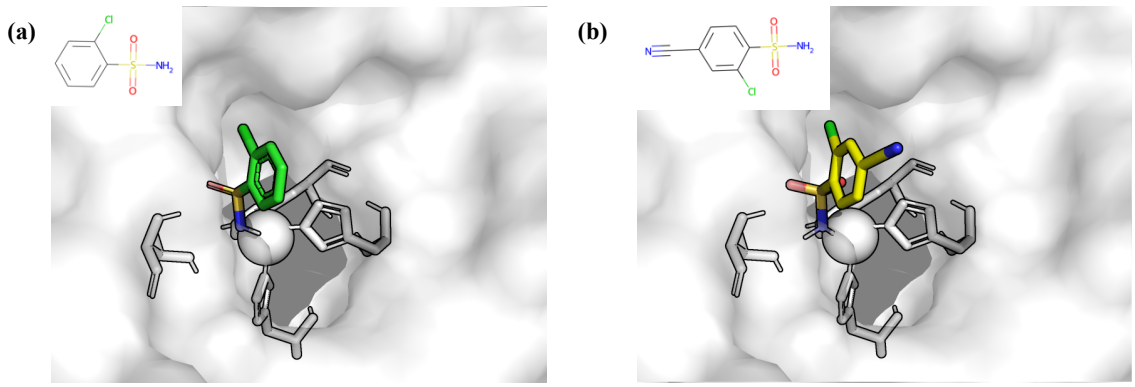


Figure 6.12: Representative 3D docked pose **(b)** for nc3D-DQN (nSuCOS) using the generated molecule (yellow) in nc3D-MolDQN with the initial starting molecule of FB1 (PDB ID: 2weh) **(a)** shown in green. H94, H96, H119, T199, and the catalytic Zn^{2+}ion are shown in white (Section 6.10.1).

Table 6.16: Average Tanimoto similarity for c3D-DQN methods computed between 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) and the corresponding generated molecule at each of the test using the reward function, LE. The values are approximated to three decimal places. Tanimoto similarity improves from 0 (least structurally similar) to 1 (most structurally similar). c3D-RxnDQN generated compounds that are more structurally similar than c3D-MolDQN and c3D-RxnMolDQN.

Method	Top 1 Average Tanimoto Similarity	Top 10 Average Tanimoto Similarity	Top 20 Average Tanimoto Similarity
c3D-MolDQN	0.834±0.145	0.647±0.115	0.592±0.103
c3D-RxnDQN	**0.985±0.058**	**0.889±0.159**	**0.852±0.187**
c3D-RxnMolDQN	0.888±0.176	0.774±0.220	0.666±0.231

In the case of c3D-DQN methods, the results for average Tanimoto similarity between the generated molecule and covalent ligands from SARS-CoV-2 main protease are shown in Table 6.16. As compared to 2D-DQN (Table 6.13) and 3D-DQN (Tables 6.14-6.15) methods, all three c3D-DQN methods generally scored higher for average Tanimoto similarity values. This is likely due to the nature of constrained covalent docking, which kept the covalent warhead of Cys-145 in the correct location in the generated molecules.

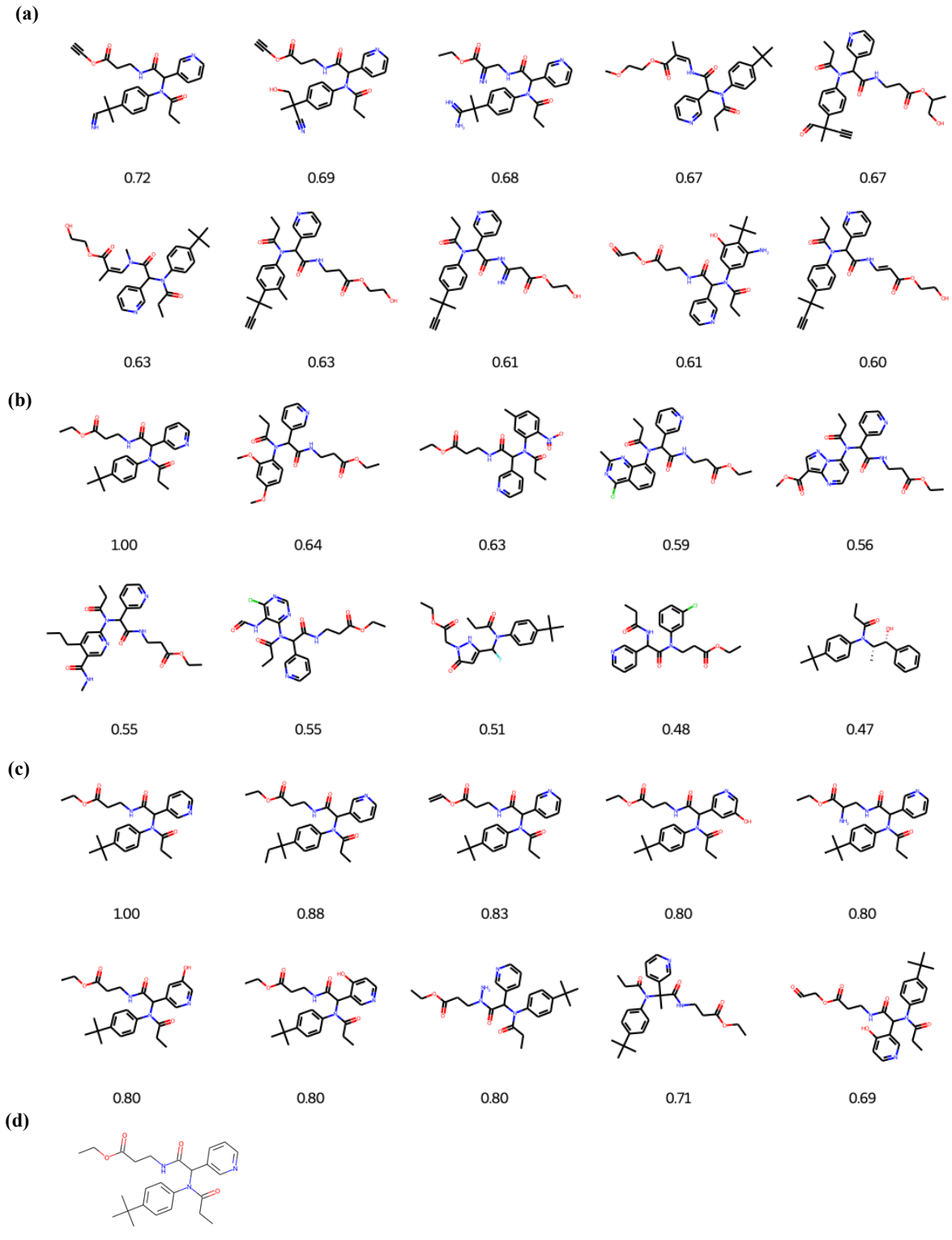


Figure 6.13: Top 10 Tanimoto similarity compounds for c3D-DQN methods computed between Mpro-x3110_0A from SARS-CoV-2 main protease **(d)** and generated molecule at each episode: **(a)** c3D-MolDQN, **(b)** c3D-RxnDQN, and **(c)** c3D-RxnMolDQN. Reaction-based methods regenerated the initial starting molecule. Hybrid c3D-RxnMolDQN generated the most structurally similar analogs.

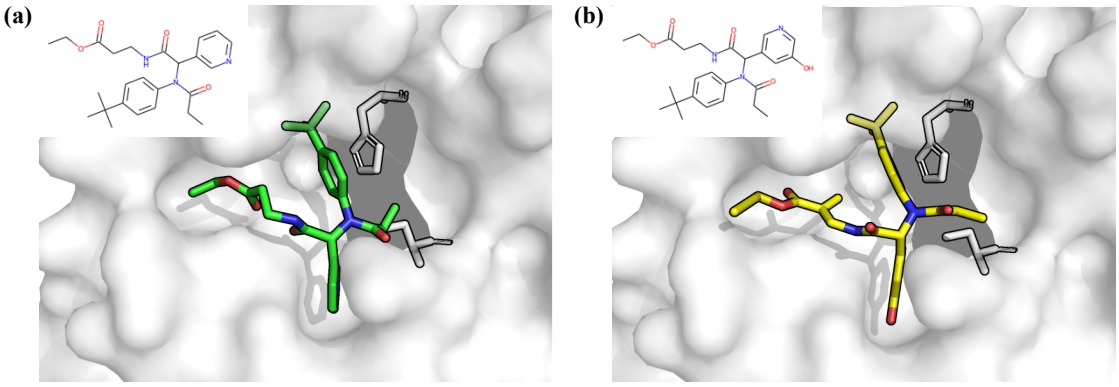


Figure 6.14: Representative 3D docked pose **(b)** for c3D-DQN methods using the generated molecule (yellow) in c3D-RxnMolDQN with the initial starting molecule of Mpro-x3110_0A in SARS-CoV-2 main protease. The catalytic dyad residues (His-41 and Cys-145) are shown in white (Section 6.10.1).

Therefore, it was rather easier for the algorithm to generate molecules that are structurally like the original covalent ligands as exemplified from the representative top 10 tanimoto similarity compounds in Figure 6.13 using Mpro-x3110_0A. This observation can also be seen from the rate of rediscovery reported in Table 6.12 and may explain the reason why the error bars shown in boxplots for synthetic accessibility in Figure 6.1(d) is narrow as compared to other types of DQN. Overall, reaction-based c3D-RxnDQN was able to design molecules that are most structurally like the desired covalent ligands, followed by c3D-RxnMolDQN and c3D-MolDQN. The representative 3D docked pose from AutoDock Vina for c3D-DQN methods is shown in Figure 6.14 using one of the generated molecules from c3D-RxnMolDQN. The remaining results from the selected covalent ligands (Appendix B, Figure B9) are included in Appendix M, Figures M16-M17.

6.11 Chemical Space Coverage

Figure 6.15 shows the per-drug relative chemical space coverage (Section 5.5.5) for 2D-DQN methods. The mean of these values for both RxnDQN (~82.2) and RxnMolDQN (~85.8) increased

by more than 4.5-fold as compared to MolDQN (~17.9). Hybrid RxnMolDQN consistently explored more chemical space than atom/bond-based MolDQN and reaction-based RxnDQN. Therefore, hybrid action space can perform better chemical space exploration than atom/bond-based and reaction-based action spaces alone.

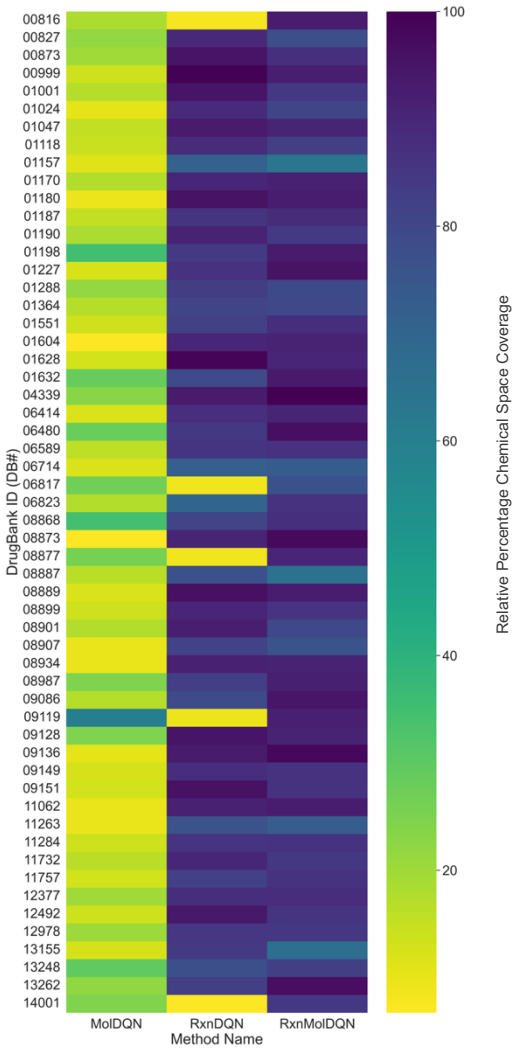


Figure 6.15: Per-drug relative percentage chemical space coverage for 2D-DQN methods computed using UMAP for 56 chemically diverse compounds from 100 DrugBank drugs (Figure 5.1) designed using the reward function, QED. The relative chemical space coverage improves from ~10% (yellow) to 100% (dark blue). In general, the reaction-based methods explore much more chemical space than atom/bond-based MolDQN.

Five DrugBank drugs where RxnDQN did not perform well on chemical space coverage (*i.e.*, yellow regions in Figure 6.15) are shown in Figure 6.16. The likely reason for poor chemical space coverage in these five cases is due to the limits imposed by available reaction centers, which reduces the number of chemical reactions that are suitable. In the raltegravir case, for example, the only appropriate reaction that can be performed by RxnDQN from the top 10 reactions is the amide reaction; the algorithm can reach an optimal QED relatively quickly and this limits the exploration of chemical space. On the other hand, RxnMolDQN overcomes this limitation by being able to perform atom/bond-based modifications whenever reaction-based modifications cannot be performed. As a result, hybrid RxnMolDQN is able to explore more chemical space than atom/bond-based (MolDQN) or reaction-based (RxnDQN) action spaces alone. All results for chemical space coverage including baselines for both QED and penalized logP as a reward function are shown in Appendix C, Figures C1 and C2, respectively.

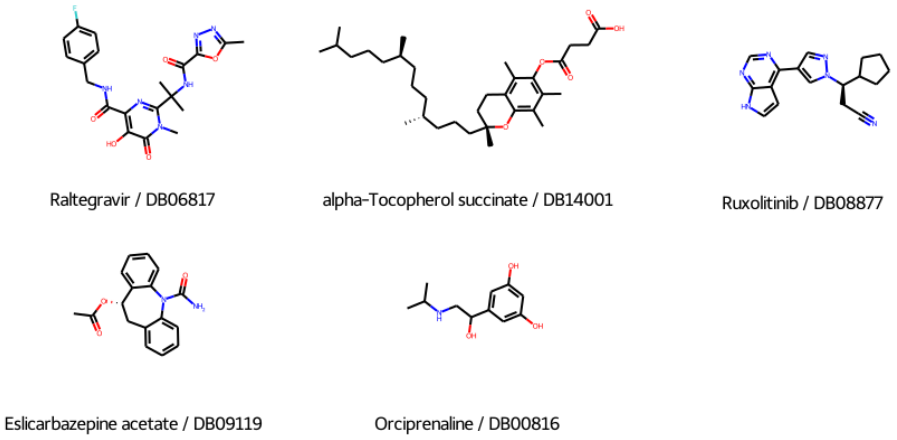


Figure 6.16: Five DrugBank (Wishart *et al.*, 2018) compounds where per-drug percentage chemical space coverage computed using UMAP scored poorly for RxnDQN.

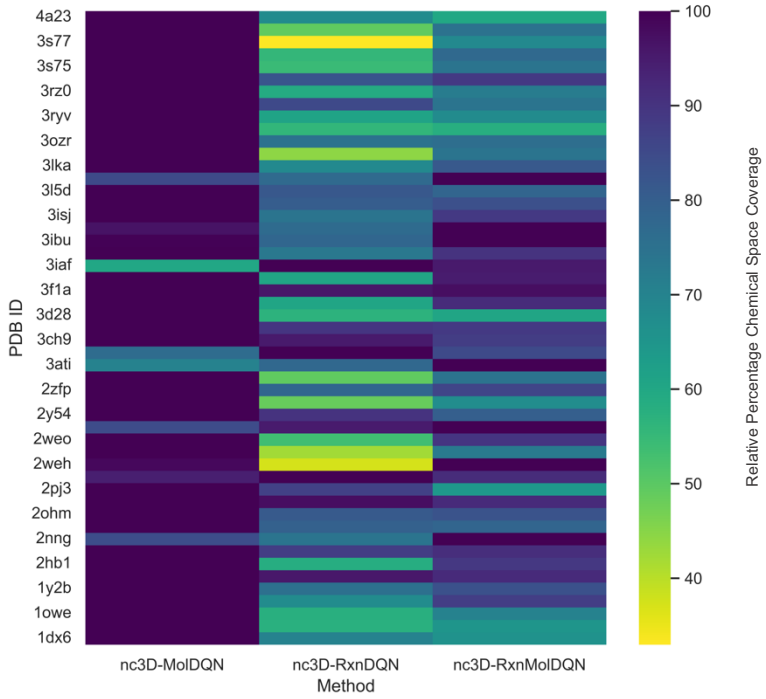


Figure 6.17: Per-ligand relative percentage chemical space coverage for nc3D-DQN (LE) methods computed using UMAP for 51 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) with the reward function, LE. The chemical space coverage improves from yellow (~35%) to dark blue (100%). In general, the chemical space coverage improves in the order of nc3D-RxnDQN, nc3D-RxnMolDQN, and nc3D-MolDQN. However, there was an indication of difference between chemical space coverage and chemical space coverage that is considered useful.

In the case of nc3D-DQN (LE) methods, the mean of per-ligand relative percentage chemical space coverage improved in the order of nc3D-RxnDQN (~71.7), nc3D-RxnMolDQN (~82.6), and nc3D-MolDQN (~97.1) as shown in Figure 6.17. This may seem a surprising result given the results shown in Figure 6.15 for 2D-DQN methods. However, even though nc3D-MolDQN explored the greatest chemical space, nc3D-MolDQN performed the worst in rediscovery (Section 6.9.2) and Tanimoto similarity (Table 6.14). These results suggest that in terms of the chemical space coverage that is considered useful, nc3D-RxnDQN performed the best despite the least chemical space coverage reported in Figure 6.17, followed by nc3D-RxnMolDQN and nc3D-RxnDQN. The usefulness of chemical space in this context is defined as at least capable of exploring a chemical space that is closer to the initial or the target molecules.

In the case of nc3D-DQN (nSuCOS) methods, the mean of the per-ligand percentage chemical space coverage increased from nc3D-RxnDQN (~74.0), to nc3D-MolDQN (~81.0), and finally nc3D-RxnMolDQN (~95.1) as shown in Figure 6.18. Therefore, unlike nc3D-DQN (LE) methods (Figure 6.17), the hybridized nc3D-RxnMolDQN explored the greatest chemical space as compared to simple atom/bond-based nc3D-MolDQN and reaction-based nc3D-RxnQN when using normalized SuCOS as the reward function.

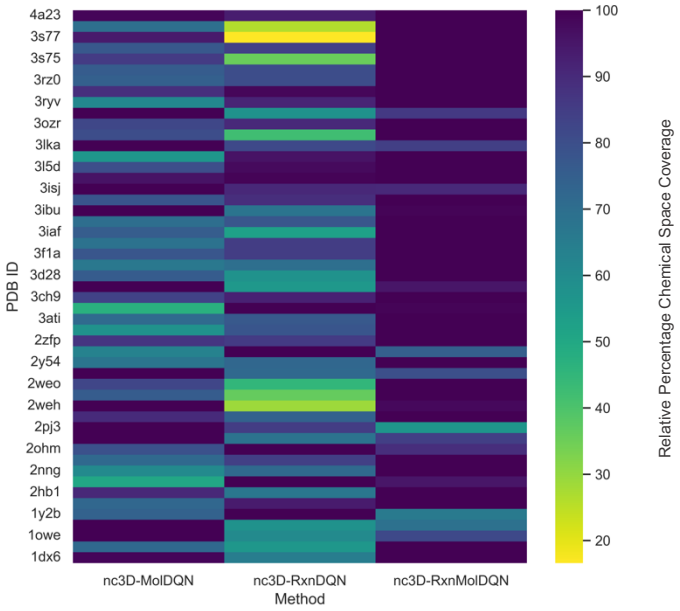


Figure 6.18: Per-ligand relative percentage chemical space coverage for nc3D-DQN (nSuCOS) methods computed using UMAP for 45 Malhotra smaller ligands (Malhotra & Karanicolas, 2017) using normalized SuCOS overlap to the larger ligand as the reward. The chemical space coverage improves from yellow (~20%) to dark blue (~100%). In general, the chemical space coverage increases from nc3D-RxnDQN to nc3D-RxnDQN to nc3D-RxnMolDQN.

For c3D-DQN methods, the mean of per-ligand relative percentage chemical space coverage improved from c3D-RxnDQN (~70.3) to c3D-MolDQN (~86.2) to c3D-RxnMolDQN (~97.6) as shown in Figure 6.19. This suggests that in the hybrid-based c3D-RxnMolDQN, it promoted the chemical space exploration by combining both atom/bond-based and reaction-based actions to make chemical modifications to a molecule. Therefore, c3D-RxnMolDQN consistently tended to

explore more chemical space than atom/bond-based c3D-MolDQN and reaction-based c3D-RxnDQN.

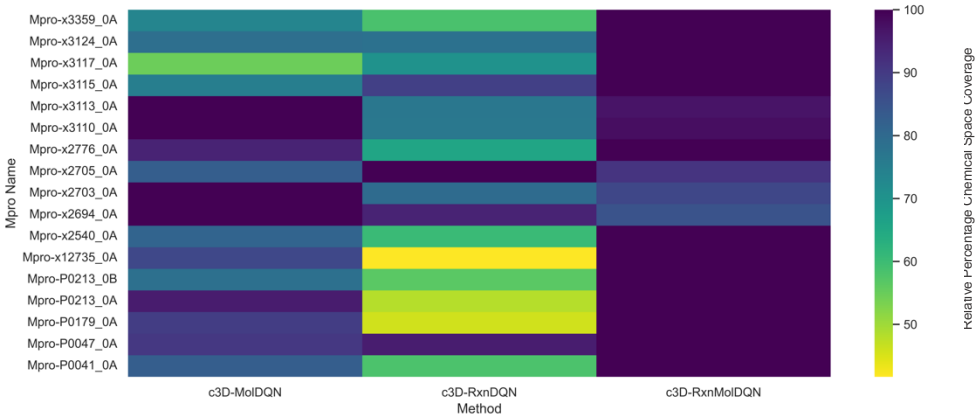


Figure 6.19: Per-ligand relative percentage chemical space coverage for c3D-DQN methods computed using UMAP for 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) with the reward function, LE. The chemical space coverage improves from yellow (45%) to dark blue (100%). In general, the chemical space coverage improves in the order of c3D-RxnDQN, c3D-MolDQN, and c3D-RxnMolDQN.

6.12 Normalized SuCOS Distribution

Table 6.17 shows the percentage of generated molecules that satisfied normalized SuCOS threshold of 0.55 (Section 5.5.12) for nc3D-DQN (LE) methods.

Table 6.17: Percentage of generated compounds that satisfy normalized SuCOS threshold (≥ 0.55) for nc3D-DQN (LE) methods computed using 51 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) with the reward function, LE. Normalized SuCOS of greater than 0.55 corresponds to less than 2Å RMSD. nc3D-MolDQN generates the most compounds that retain the shape and feature overlaps as compared to reaction-based methods of nc3D-RxnDQN and nc3D-RxnMolDQN.

Method	**Normalized SuCOS ≥ 0.55 (%)**
nc3D-MolDQN	**8.42**
nc3D-RxnDQN	5.24
nc3D-RxnMolDQN	2.03

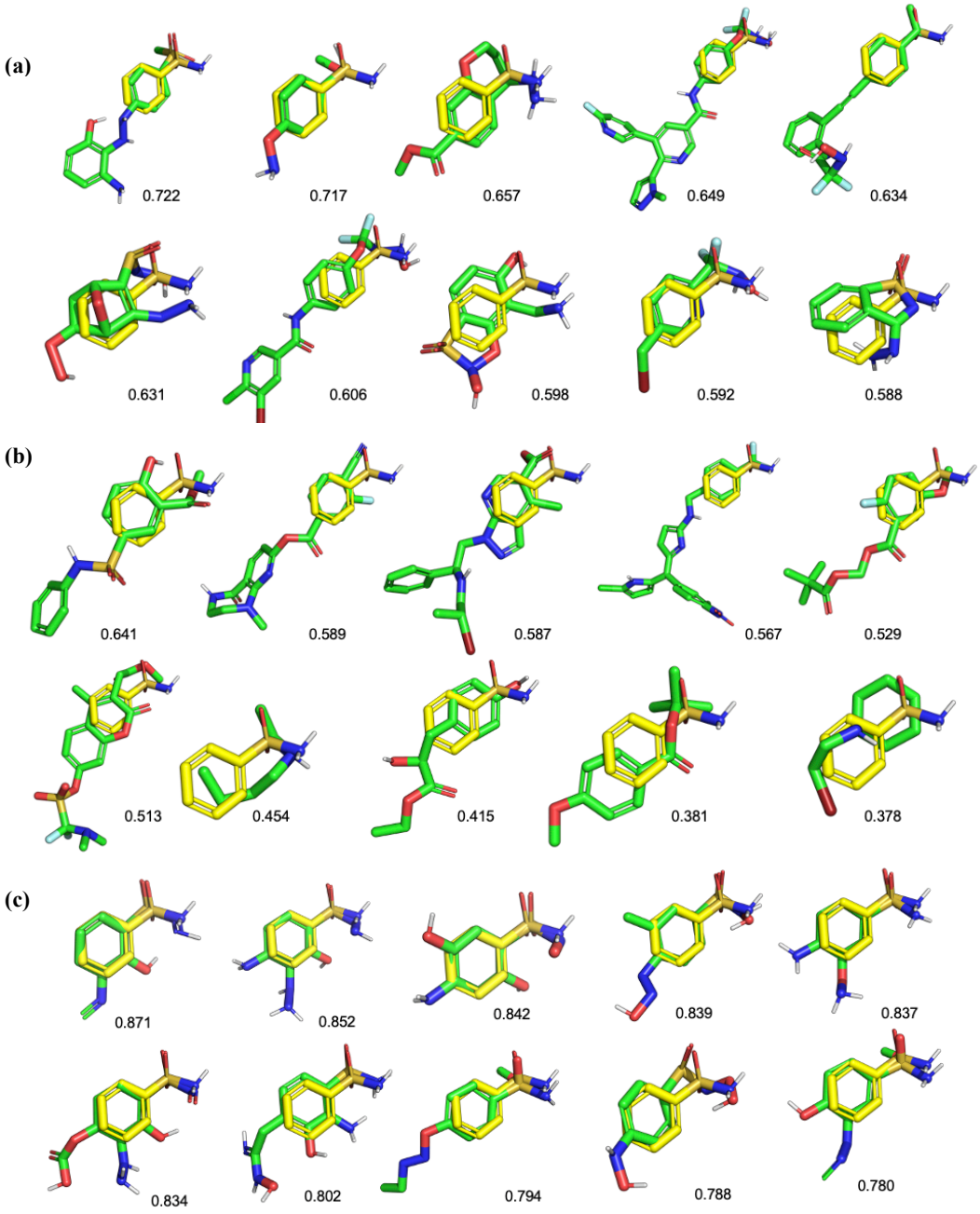


Figure 6.20: Top 10 normalized SuCOS computed between FB2/2wej (yellow) from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule (green) at each episode for nc3D-DQN (LE) methods: **(a)** nc3D-MolDQN, **(b)** nc3D-RxnDQN, and **(c)** nc3D-RxnMolDQN. Oxygen, nitrogen, sulphur, and bromine atoms are shown in red, blue, mustard, and dark red, respectively. All methods generated molecules with nSuCOS > 0.55.

When comparing the results for the reaction-based methods, nc3D-RxnDQN generated 2.6 times more molecules that satisfied normalized SuCOS threshold than nc3D-RxnMolDQN. This

suggests that when using the hybrid-based action space, it is harder to retain the shape and feature overlaps as compared to atom/bond-based action space or reaction-based action space alone. The representative top 10 normalized SuCOS results for nc3D-DQN (LE) methods are shown in Figure 6.20. All nc3D-DQN (LE) methods were capable of proposing compounds that satisfied nSuCOS threshold when optimizing for predicted ligand efficiency.

Table 6.18: Percentage of generated compounds that satisfy normalized SuCOS threshold (≥ 0.55) for nc3D-DQN (nSuCOS) methods computed using 45 Malhotra smaller ligands (Malhotra & Karanicolas, 2017) with the reward function, nSuCOS. Normalized SuCOS of greater than 0.55 corresponds to less than 2Å RMSD. nc3D-MolDQN generates the most compounds that retain the shape and feature overlaps as compared to reaction-based methods.

Method	Normalized SuCOS ≥ 0.55 (%)
nc3D-MolDQN	**6.71**
nc3D-RxnDQN	3.17
nc3D-RxnMolDQN	2.85

In the case of nc3D-DQN (nSuCOS) methods, nc3D-MolDQN generated the greatest proportion of molecules that satisfied normalized SuCOS threshold (Table 6.18), which is a similar observation as nc3D-DQN (LE) methods in Table 6.17. Despite using normalized SuCOS as the reward, generated ligands that satisfied this threshold are below the values reported in Table 6.17 except for nc3D-RxnMolDQN. This may suggest that single-objective optimization with normalized SuCOS is difficult when using nc3D-DQN methods. However, when comparing the results in Tables 6.17-6.18, the performance is better for nc3D-RxnMolDQN with normalized SuCOS as the reward (Tables 6.17 and 6.18). This indicates that it is likely to be easier to optimize for normalized SuCOS in the hybridized action space as compared to the reward function of predicted ligand efficiency.

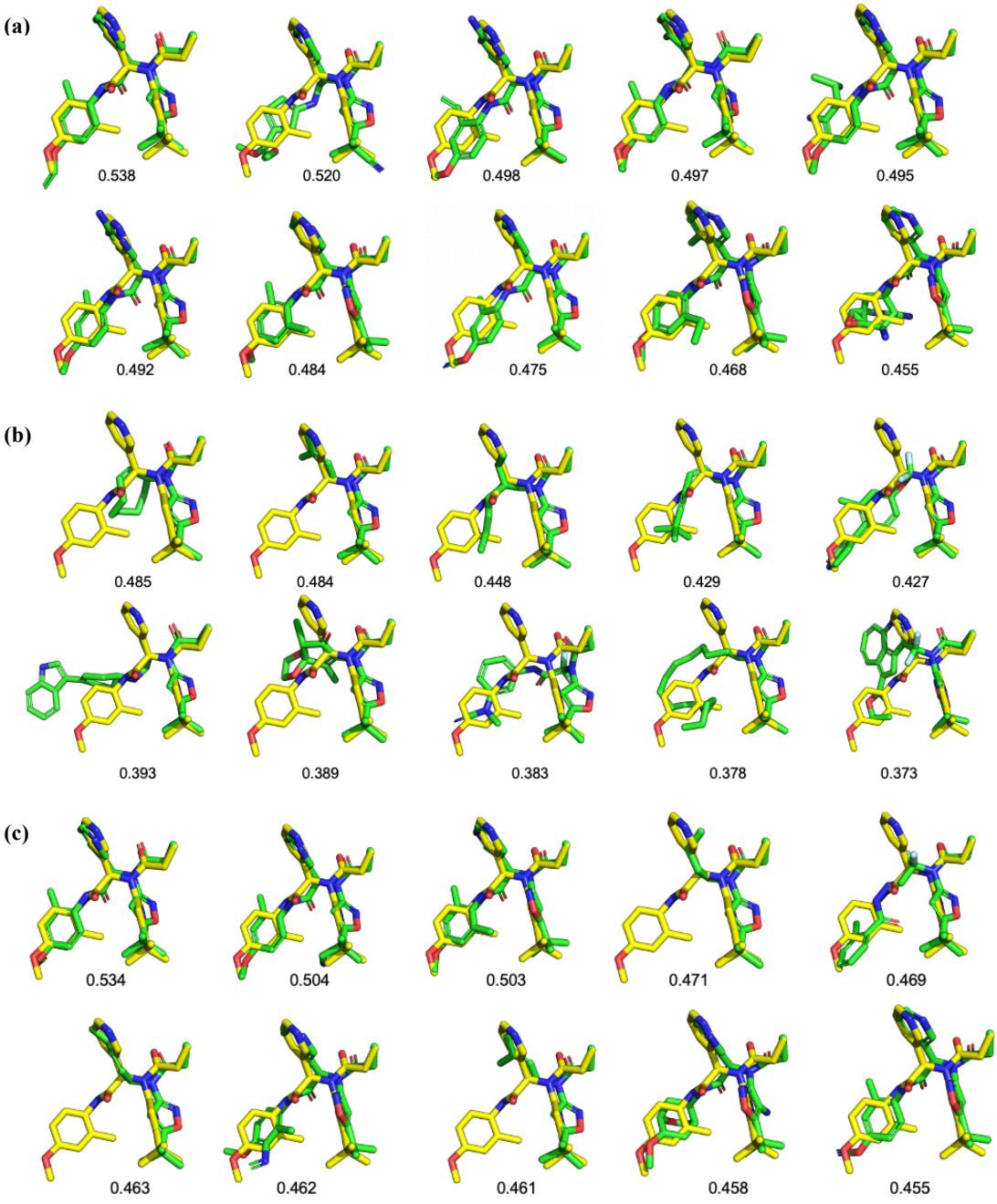


Figure 6.21: Top 10 designed ligands and their normalized SuCOS values for c3D-DQN methods computed between Mpro-x2694_0A (yellow) from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) and the generated molecule (green): **(a)** c3D-MolDQN, **(b)** c3D-RxnDQN, and **(c)** c3D-RxnMolDQN. Oxygen and nitrogen atoms are shown in red and blue, respectively. c3D-MolDQN generated molecules that are relatively close to nSuCOS of 0.55.

Table 6.19: Percentage of generated compounds that satisfy normalized SuCOS threshold (≥ 0.55) for c3D-DQN methods computed using 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) with the reward function, LE. Normalized SuCOS of greater than 0.55 corresponds to less than 2Å RMSD. c3D-RxnDQN generates the most compounds that retain the shape and feature overlaps as compared to c3D-MolDQN and c3D-RxnMolDQN.

Method	Normalized SuCOS ≥ 0.55 (%)
c3D-MolDQN	0.023
c3D-RxnDQN	**0.027**
c3D-RxnMolDQN	0.023

For c3D-DQN methods, c3D-RxnDQN generated the greatest proportion of molecules that satisfied the normalized SuCOS threshold as compared to c3D-MolDQN and c3D-RxnMolDQN as shown in Table 6.19. However, in general, given the low occupancy of the docked poses that satisfied this threshold, c3D-DQN methods are not well suited to retain the shape and feature overlaps of the original covalent ligands. This may be one of the limitations in using constrained docking, where we have more confidence in the docked location, but whether this leads to retaining pharmacophoric-based features that could be measured by normalized SuCOS is another question. Representative top 10 normalized SuCOS compounds for the three methods are shown in Figure 6.21. All c3D-DQN methods were able to generate compounds that are relatively close to nSuCOS threshold, but not to the same degree as Figure 6.20 for nc3D-DQN (LE) methods. When comparing c3D-DQN and nc3D-DQN (LE) methods, nc3D-DQN methods satisfied nSuCOS threshold at least ~90 times more across all action spaces. Therefore, this further suggests the difficulty in satisfying the nSuCOS threshold for constrained docking as compared to non-constrained docking when using LE as the reward.

6.13 Silly Walks

Figure 6.22 shows the average silly walks (Section 5.5.6) for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods with varying reward functions and docking methods (if relevant): 2D-DQN (QED), nc3D-DQN (LE), nc3D-DQN (nSuCOS), and c3D-DQN (LE) methods. For MolDQN-based methods (blue box plots), the median of average silly score ranges from ~0.35 to 0.5. For reaction-based methods, this value ranges from ~0.23 to 0.26 for RxnDQN-based methods (orange box plots) and ~0.25 to 0.37 for RxnMolDQN-based methods (green box plots). Therefore, reaction-based methods generated fewer "silly" molecules than MolDQN-based methods regardless of the type of reward functions (2D and 3D-based) and docking (non-constrained or constrained). When comparing the 2D-DQN and 3D-DQN methods, 3D-DQN generated fewer silly molecules as compared to 2D-DQN for MolDQN-based methods. In the case of RxnDQN-based methods, this degree is quite similar across 2D-DQN and 3D-DQN methods. For RxnMolDQN-based methods, 2D-DQN method generated fewer silly molecules than 3D-DQN methods. Henceforth, the impact of each action space to 2D-DQN and 3D-DQN methods in terms of silly walks depends on the action space being considered.

It is also interesting to note that the trends observed in Figure 6.22 for silly walks correlates with synthetic accessibility results reported in Figure 6.1. Consequently, it is expected and reasonable to conclude that methods that are synthetically accessibility also generate fewer silly molecules. Silly walks metric further strengthens the fact that the reaction-based methods are more suitable for *de novo* drug design as compared to MolDQN-based methods.

For 2D-DQN methods (Table 5.2), all results for silly walks including baselines using QED and penalized logP as a reward function are shown in Appendix H, Figure H1-H4.

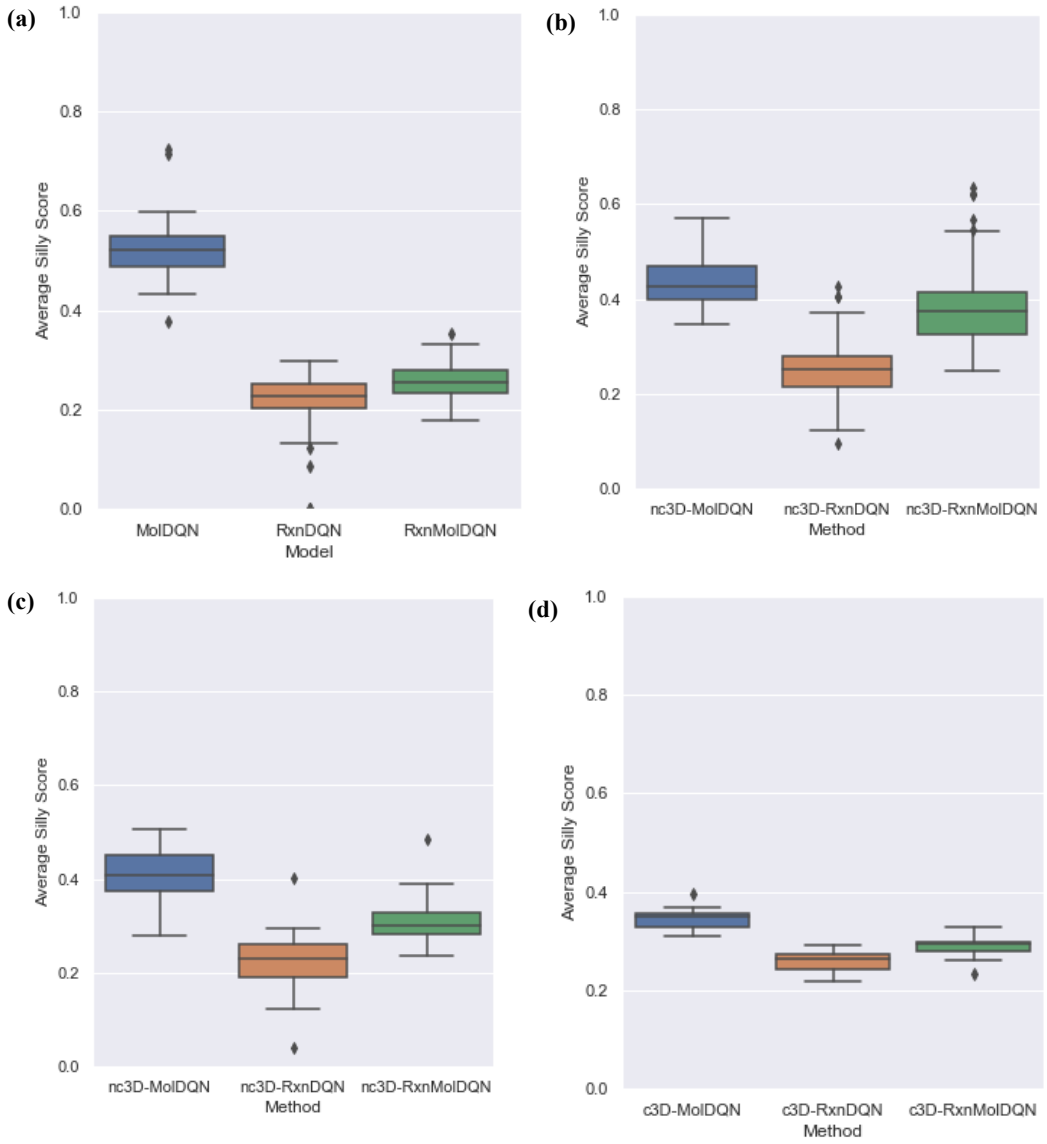


Figure 6.22: Average silly score for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods with respective reward function in **(a)** 2D-DQN (QED), **(b)** nc3D-DQN (LE), **(c)** nc3D-DQN (nSuCOS), and **(d)** c3D-DQN (LE). These values are computed using the dataset described in Figure 5.1 for **(a)**, Figure 5.2 for **(b)-(c)**, and Figure 5.3 for **(d)**. Silly walks score improves from 1 (most silly compound) to 0 (least silly compound). Across varying methods with different reward functions, reaction-based methods generated fewer silly molecules than the atom/bond-based methods (MolDQN, nc3D-MolDQN, and c3D-MolDQN).

6.14 Optimization Efficiency

6.14.1 2D-DQN Methods

Figure 6.23 shows the reward progression (Section 5.5.8) for 2D-DQN methods. Based on Figures 6.23(b)-(c), the reward progression for the reaction-based methods consists of three stages: (1) learning for the first 500 episodes; (2) improving reward based on stage 1 for 500-1,500th episodes; (3) starting to reach an optimal reward between 1,500-3,000th episodes.

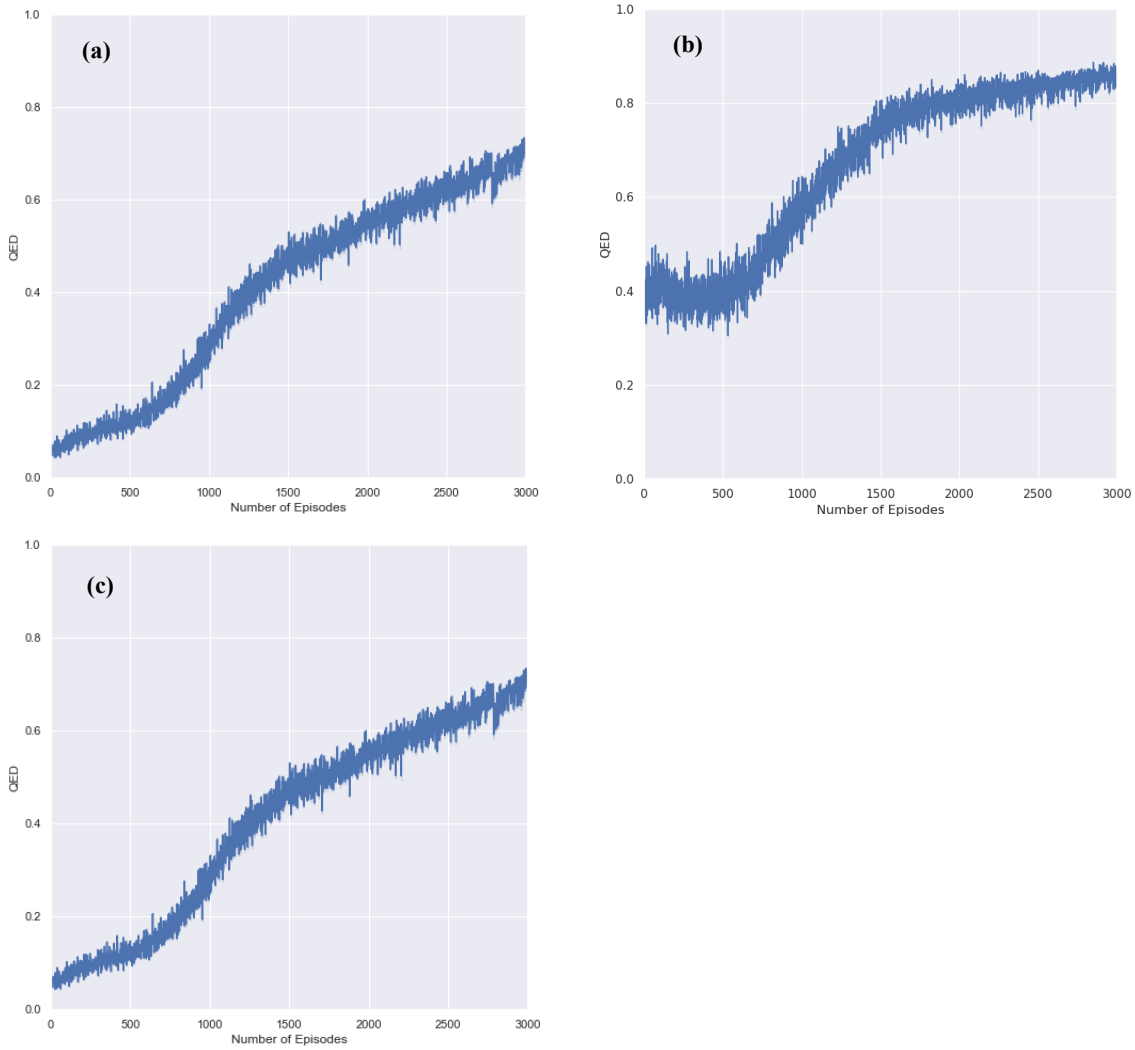


Figure 6.23: Reward progression for 2D-DQN methods computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED: **(a)** MolDQN, **(b)** RxnDQN, and **(c)** RxnMolDQN. All rewards from each DrugBank drugs are plotted for the corresponding number of episodes. Progression of reward for the reaction-based methods consist of three stages, whereas the progression is almost linear for MolDQN.

MolDQN showed consistent improvement in the reward, but there was no indication of plateauing for an optimal reward by 3,000th episodes (Figure 6.23(a)). Therefore, all three methods achieved higher reward when running for more episodes, but reaction-based methods reached an optimal reward twice as fast (*i.e.*, by 1,500th episode) and generated compounds with higher QED values than MolDQN. All results for optimization efficiency for both QED and penalized logP as a reward function are shown in Appendix J, Figures J1-J3 and J4-J6, respectively.

6.14.2 nc3D-DQN (LE) Methods

Figure 6.24 shows the results for average reward progression (Section 5.5.8) for nc3D-DQN (LE) methods. The reward progression for the reaction-based methods consists of three stages: (1) learning for the first 40 episodes; (2) improving reward based on stage 1 for 40-150th episodes; (3) starting to reach an optimal reward between 150-300th episodes for nc3D-RxnDQN (Figure 6.24(b)) and leveling out of reward between 150-300th episodes for nc3D-RxnMolDQN (Figure 6.24(c)). In the case of nc3D-MolDQN (Figure 6.24(a)), the progression of reward is an almost linear horizontal line. This suggests that nc3D-MolDQN either required longer episodes than the reaction-based methods to conduct the learning or the improvement in the reward after learning during the first 40 episodes is not as significant as compared to the reaction-based methods. These results are interesting since running for longer number of episodes did not necessary lead to a higher reward, especially for nc3D-MolDQN and nc3D-RxnMolDQN. The results for nc3D-RxnMolDQN are likely caused due to the tendency to generate larger molecules (*i.e.*, a greater number of heavy atoms, leading to lower predicted ligand efficiency) after reaching the peak in predicted ligand efficiency, since the method decided to make more chemical modifications across the number of episodes. In fact, across all three methods, 150 episodes seem sufficient to

reach an optimal reward. In terms of a smaller number of episodes to reach an optimal reward, nc3D-RxnMolDQN performed the best.

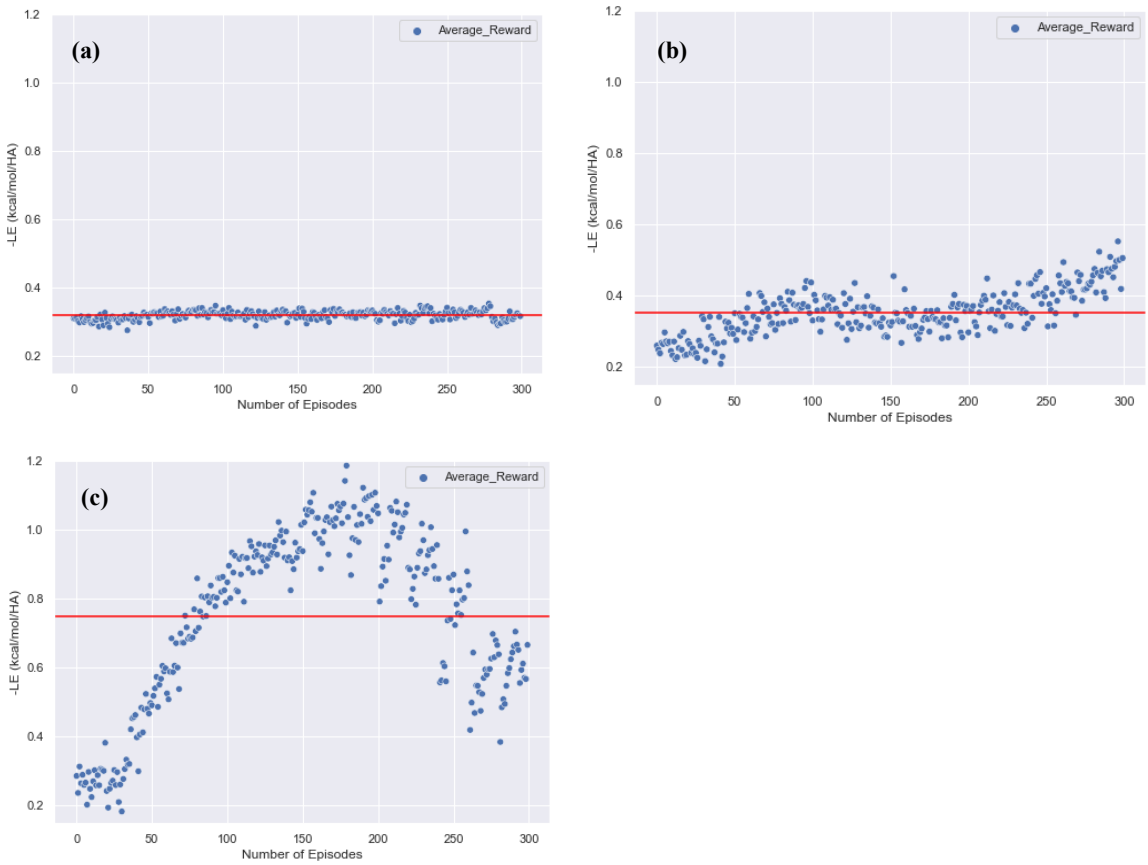


Figure 6.24: Reward progression for nc3D-DQN (LE) methods computed using average reward from 51 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) for each episode with the reward function, LE: **(a)** nc3D-MolDQN, **(b)** nc3D-RxnDQN, and **(c)** nc3D-RxnMolDQN. Red horizontal bars represent the average of reward values for each method (nc3D-MolDQN: 0.320; nc3D-RxnDQN: 0.352; nc3D-RxnMolDQN: 0.749). Progression of reward for the reaction-based methods consist of three stages, whereas the progression is almost linear for nc3D-MolDQN.

6.14.3 nc3D-DQN (nSuCOS) Methods

Figure 6.25 shows the results for average reward progression (Section 5.5.8) for nc3D-DQN (nSuCOS) methods. For nc3D-DQN (nSuCOS) methods, the pharmacophoric overlap of the ligand as it is being designed with the desired active compound (*i.e.*, Malhotra larger ligands)

improves gradually and starts to reach an optimal reward between 150-300th episodes. For all variants of nc3D-DQN (nSuCOS) methods, 150 episodes seems to be sufficient to reach an optimal reward. In terms of the best method to optimize the single objective reward function, which is defined by the slope between 40-150th episodes, nc3D-MolDQN performed the best, followed by nc3D-RxnMolDQN and nc3D-RxnDQN. These results suggest that it is easier to optimize for normalized SuCOS with atom/bond-based actions, as discussed in Section 6.7.

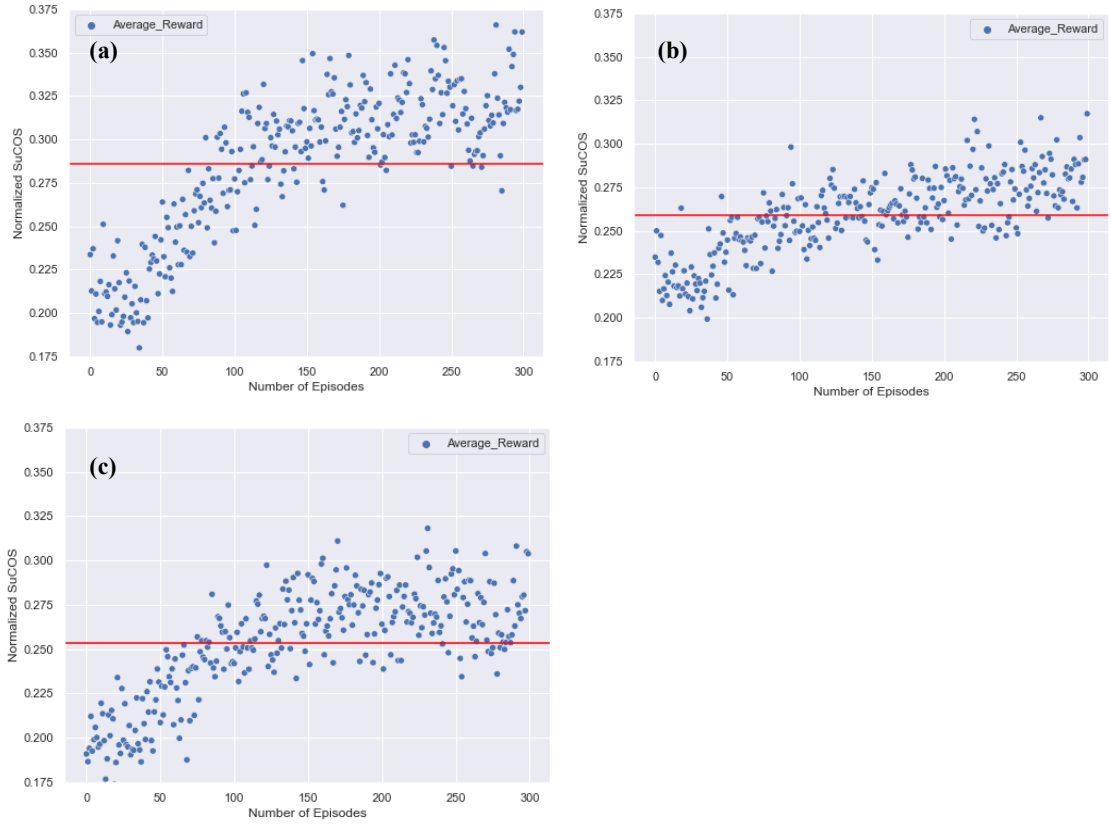


Figure 6.25: Reward progression for nc3D-DQN (nSuCOS) methods computed using average reward from 45 Malhotra smaller ligands (Malhotra & Karanicolas, 2017) for each episode with the reward function, normalized SuCOS: **(a)** nc3D-MolDQN, **(b)** nc3D-RxnDQN, and **(c)** nc3D-RxnMolDQN. Red horizontal bars represent the average of reward values for each method (nc3D-MolDQN: 0.286; nc3D-RxnDQN: 0.259; nc3D-RxnMolDQN: 0.254). nc3D-MolDQN performed better for reward optimization of nSuCOS as compared to the reaction-based methods.

6.14.4 c3D-DQN Methods

Figure 6.26 shows the results for average reward progression (Section 5.5.8) for c3D-DQN methods. Based on Figures 6.26(a)-(c), the reward progression consists of three stages: (1) learning for the first 40 episodes; (2) improving reward based on stage 1 for 40-150th episodes; (3) starting to reach an optimal reward between 150-300th episodes. Therefore, for c3D-DQN methods, 150 episodes seem sufficient to reach an optimal reward.

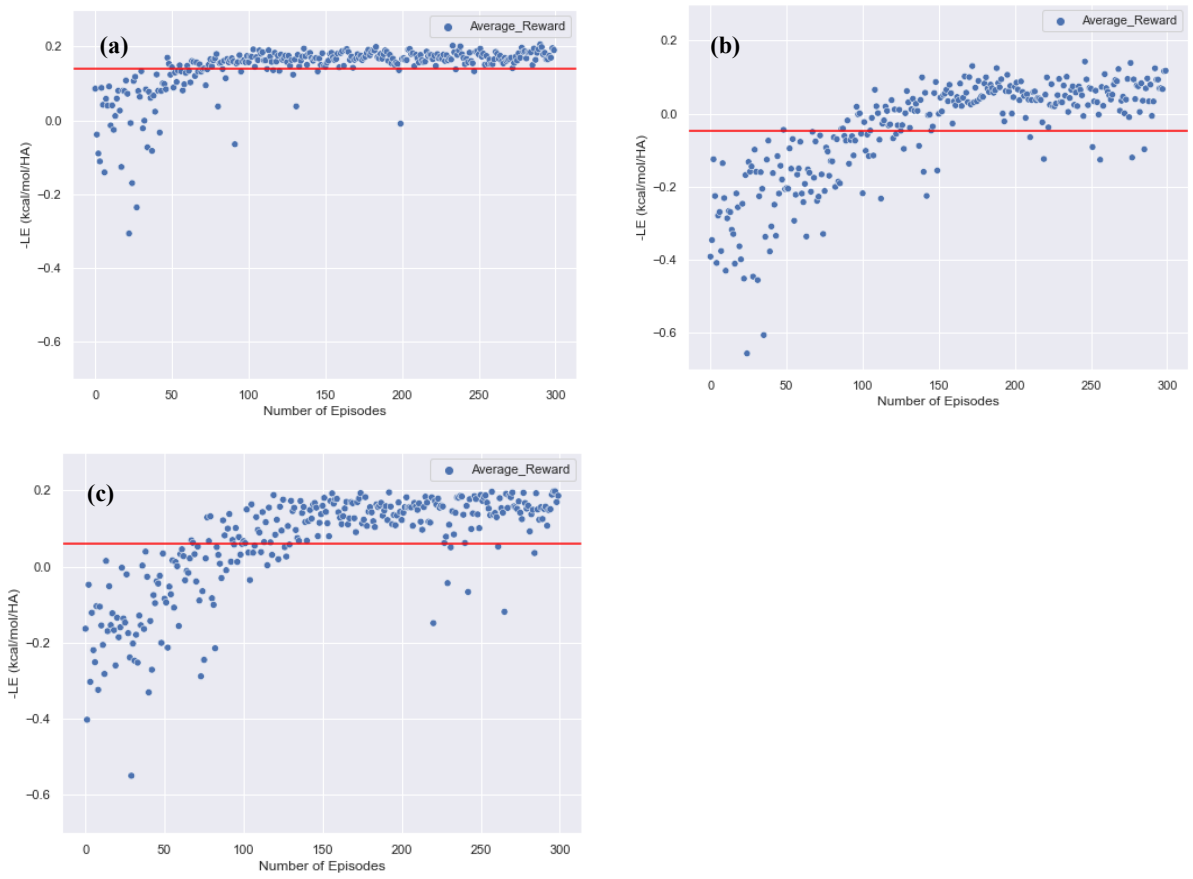


Figure 6.26: Reward progression for c3D-DQN methods computed using the average of 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) for each episode with the reward function, LE: **(a)** c3D-MolDQN, **(b)** c3D-RxnDQN, and **(c)** c3D-RxnMolDQN. Red horizontal bars represent the average of reward values for each method (c3D-MolDQN: 0.140; c3D-RxnDQN: -0.0479; c3D-RxnMolDQN: 0.0608). The stages in which the rewards progress for the c3D-DQN methods are similar.

The negative LE values are likely due to the initial ligand designs clashing with the binding pocket by making larger chemical modifications as discussed earlier (Section 6.6.2). In terms of optimization efficiency, which is defined by the average LE in this case, c3D-MolDQN performed the best followed by c3D-RxnMolDQN and c3D-RxnDQN.

6.15 Uniqueness

Tables 6.20 and 6.21 show the results for average percentage uniqueness (Section 5.5.9) for 2D-DQN and 3D-DQN methods, respectively. In general, MolDQN-based methods performed better in average uniqueness than reaction-based methods regardless of the types of DQN. However, despite moving from atom/bond-based action space (MolDQN-based methods) to reaction-based action space (RxnDQN-based methods) where chemical modifications may be considered restricted based on the number of reagents and reactions, the results show that the RxnDQN-based methods can generate unique molecules. The hybrid RxnMolDQN-based methods achieved an even higher degree of uniqueness whilst exploring more chemical space in most cases (Section 6.11). Therefore, the compensation in uniqueness for reaction-based methods is arguably worth the cost, since MolDQN-based methods' synthetic accessibility is not guaranteed (Figure 6.1).

Table 6.20: Average percentage uniqueness of the molecules generated by the first 1,500 episodes for 2D-DQN methods computed using 100 chemically diverse drugs from DrugBank (Wishart *et al.*, 2018) with the reward function, QED. The values are reported to one decimal place. Average of molecules generated in the top 1,500 episodes was considered since RxnDQN and RxnMolDQN are approximately twice as efficient in reaching the optimal reward than MolDQN (Section 6.14.1). Hybridizing the action space of RxnDQN with that of MolDQN improves the performance over RxnDQN in RxnMolDQN.

Method	Average Uniqueness for Top 1,500 Episodes (%)
MolDQN	**99.9±0.1**
RxnDQN	83.7±20.1
RxnMolDQN	93.5±3.2

Table 6.21: Average percentage uniqueness of the molecules generated by the first 150 episodes for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods in 3D-DQN: nc3D-DQN (LE), nc3D-DQN (nSuCOS), and c3D-DQN (LE). These values were computed using the dataset described in Section 5.2 and reported to one decimal place. Average of molecules generated in the top 150 episodes was considered since 3D-DQN methods were found to reach an optimal reward by the first 150 episodes. Uniqueness mostly improved in the order of RxnDQN-based, RxnMolDQN-based, and MolDQN-based methods.

Method	Reward	Average Uniqueness for Top 150 Episodes in MolDQN-based Method (%)	Average Uniqueness for Top 150 Episodes in RxnDQN-based Method (%)	Average Uniqueness for Top 150 Episodes in RxnMolDQN-based Method (%)
nc3D-DQN	LE	**98.1±1.8**	67.7±18.0	74.4±9.3
nc3D-DQN	nSuCOS	95.8±3.8	74.2±25.5	**97.1±3.8**
c3D-DQN	LE	**97.9±2.2**	80.3±10.8	94.6±4.6

When comparing 2D-DQN methods to nc3D-DQN (LE) methods, the results are similar except for the reaction-based methods that differ by more than 20%. This suggests that reaction-based nc3D-DQN (LE) methods are more likely to generate duplicate molecules than the reaction-based 2D-DQN methods. This is likely due to the difference in the nature of the reward where predicted ligand efficiency limits the chemical space exploration than QED. In the case of comparison between 2D-DQN methods and nc3D-DQN (nSuCOS) methods, the average uniqueness scores are similar across same type of action spaces. Therefore, the reward function of normalized SuCOS avoids the generation of duplicated molecules to the same extent as the reward function of QED. Consequently, the reward of normalized SuCOS promoted more unique exploration of chemical space than predicted ligand efficiency for nc3D-DQN methods.

Like the nc3D-DQN (nSuCOS) methods, the average uniqueness values for c3D-DQN methods are similar to 2D-DQN methods. This result is interesting since despite of using constrained docking to keep the covalent warhead of Cys-145 for SARS-CoV-2 main protease, c3D-DQN methods generated more proportion of unique molecules than nc3D-DQN (LE) methods. This

stresses the fact that chemical space exploration is performed sufficiently regardless of the docking types (*i.e.*, constrained or non-constrained docking).

For 2D-DQN methods, all results for uniqueness including baselines using QED and penalized logP as a reward function are shown in Appendix K, Tables K1 and K2, respectively.

6.16 Computational Efficiency

Table 6.22 shows the results for computational efficiency (Section 5.5.10) for 2D-DQN and 3D-DQN methods. For 2D-DQN methods, the average computational efficiency increased by more than 70% for reaction-based methods as compared to MolDQN. Therefore, despite running for double the episodes required to reach an optimal reward (Section 6.14.1), reaction-based methods are more computationally efficient than MolDQN. This is likely because the reaction-based action space is more computationally efficient than the atom/bond-based action space. All results for computational efficiency in 2D-DQN methods including baselines using QED and penalized logP as a reward function are shown in Appendix L, Tables L1 and L2, respectively.

Table 6.22: Computational efficiency for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods in 2D-DQN, nc3D-DQN (LE), nc3D-DQN (nSuCOS), and c3D-DQN computed using the relevant dataset (Section 5.2). The values are approximated to one decimal place. Reaction-based methods are more computationally efficient than MolDQN-based methods.

Method	Reward	Average Time for MolDQN-based Method (hr)	Average Time for RxnDQN-based Method (hr)	Average Time for RxnMolDQN-based Method (hr)
2D-DQN	QED	23.7±8.8	**13.1±4.8**	13.9±4.3
nc3D-DQN	LE	3.6±2.4	**1.7±0.6**	3.3±1.1
nc3D-DQN	nSuCOS	3.6±2.7	**2.7±0.9**	8.2±2.5
c3D-DQN	LE	13.7±3.8	**3.8±0.9**	13.3±1.6

For nc3D-DQN methods, RxnDQN-based methods are generally twice as efficient compared to MolDQN-based and RxnMolDQN-based methods. This result is expected since RxnDQN-based methods perform 5 steps per episode, whereas MolDQN-based and RxnMolDQN-based methods perform 10 steps per episode. Despite running for the same number of steps per episode, nc3D-RxnMolDQN is more efficient than nc3D-MolDQN when optimizing for LE and vice-versa when optimizing for nSuCOS. For c3D-DQN methods, the average time for various action spaces is longer as compared to nc3D-DQN methods due to the extra docking procedures involved in AutoDock Vina to perform the constrained docking (Section 4.3.1). c3D-RxnDQN is an efficient method since it is more than 3.5-fold computationally less expensive as compared to c3D-MolDQN and c3D-RxnMolDQN even when considering the number of steps per episode (*i.e.*, 5 steps in c3D-RxnDQN versus 10 steps in c3D-MolDQN and c3D-RxnMolDQN).

6.17 Lipinski's Rule of Five

Table 6.23 shows the results for Lipinski's rule of five (Section 5.5.13) for 3D-DQN methods with its corresponding results from 2D-DQN methods (Table 6.24) using the same dataset (Section 5.2).

Table 6.23: Lipinski's rule of five (LRo5) for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods in 3D-DQN: nc3D-DQN (LE), nc3D-DQN (nSuCOS), and c3D-DQN (LE). These values were computed using the dataset described in Section 5.2 and reported to three significant figures. These methods tend to generate compounds that satisfied LRo5 approximately in the range of 80-90%.

Method	Reward	Average Percentage of Passed LRo5 for MolDQN-based Method (%)	Average Percentage of Passed LRo5 for RxnDQN-based Method (%)	Average Percentage of Passed LRo5 for RxnMolDQN-based Method (%)
nc3D-DQN	LE	83.6±17.7	91.9±10.4	**93.2±3.46**
nc3D-DQN	nSuCOS	83.2±20.3	**88.9±9.17**	82.2±10.4
c3D-DQN	LE	**88.1±6.64**	76.5±8.99	80.8±6.85

For nc3D-DQN methods, the reaction-based methods tend to generate more compounds that satisfied this rule than nc3D-MolDQN regardless of the reward function. This suggests that the molecular generation governed by the reaction-based action space is producing a positive impact towards molecular properties to satisfy the Lipinski's rule of five.

For c3D-DQN methods (Table 6.23), c3D-MolDQN generated more compounds that satisfied Lipinski's rule of five on average as compared to the reaction-based methods. This result may seem unexpected given the constant high performances reported by the reaction-based methods on properties such as synthetic accessibility (Section 6.3), Walters' compound quality (Section 6.8), and silly walks score (Section 6.13), but out of these metrics, the Lipinski's rule of five considers the most elementary and simple factors. Therefore, even though c3D-MolDQN performed better on average as compared to the reaction-based methods on this rule, this is one of many metrics, and the quality of molecules generated by the reaction-based methods are still better than c3D-MolDQN.

Table 6.24 Lipinski's rule of five (LRo5) for MolDQN-based, RxnDQN-based, and RxnMolDQN-based methods in 2D-DQN (QED) using the same dataset as the corresponding method from Table 6.23. The values are approximated to three significant figures. All methods generated compounds that satisfied LRo5 to the same degree ranging from approximately 92-97%.

Method	Dataset Type	Average Percentage of Passed LRo5 for MolDQN-based Method (%)	Average Percentage of Passed LRo5 for RxnDQN-based Method (%)	Average Percentage of Passed LRo5 for RxnMolDQN-based Method (%)
2D-DQN (QED)	nc3D-DQN (LE)	**96.9±4.25**	94.6±2.54	93.1±1.94
2D-DQN (QED)	nc3D-DQN (nSuCOS)	**97.1±2.90**	94.7±2.60	93.3±1.94
2D-DQN (QED)	c3D-DQN (LE)	**96.7±2.32**	92.2±1.98	92.2±1.18

For 2D-DQN methods (Table 6.24), the average percentage of compounds that satisfied Lipinski's rule of five is higher as compared to 3D-DQN methods (Table 6.23) with the same

dataset. This confirms that the reward function of QED (Section 2.2.11) is better suited for generating compounds that satisfy this rule as compared to 3D-based reward functions such as predicted ligand efficiency (Section 3.2.5) and normalized SuCOS (Section 3.2.7). Although MolDQN-based methods reported the highest average percentage for Lipinski's rule of five in 2D-DQN methods, the results are almost identical between reaction-based methods. The standard deviation for the average Lipinski's rule of five tend to be larger for MolDQN-based methods as compared to the reaction-based methods in both 2D-DQN (Table 6.24) and 3D-DQN (Table 6.25) methods. This indicates that MolDQN-based methods generated compounds with a wider range in quality as compared to reaction-based methods.

6.18 Multi-objective Optimization

6.18.1 2D-DQN Methods

Figure 6.27 shows the distribution of QED values for last 20 molecules in each 100 DrugBank drugs in the context of multi-objective optimization (Section 5.5.14) between QED and Tanimoto similarity to the initial drug for 2D-DQN methods. The performance in QED improved from MolDQN to RxnDQN to RxnMolDQN for ω = 0.0, 0.2, and 0.4 in Figures 6.27(a)-(c). Although there was no significant difference in the median across these methods after ω = 0.6, the median QED values for reaction-based methods are higher as compared to MolDQN when reaching ω = 1.0. In this multi-objective optimization case, RxnDQN and RxnMolDQN are a better optimization method than MolDQN.

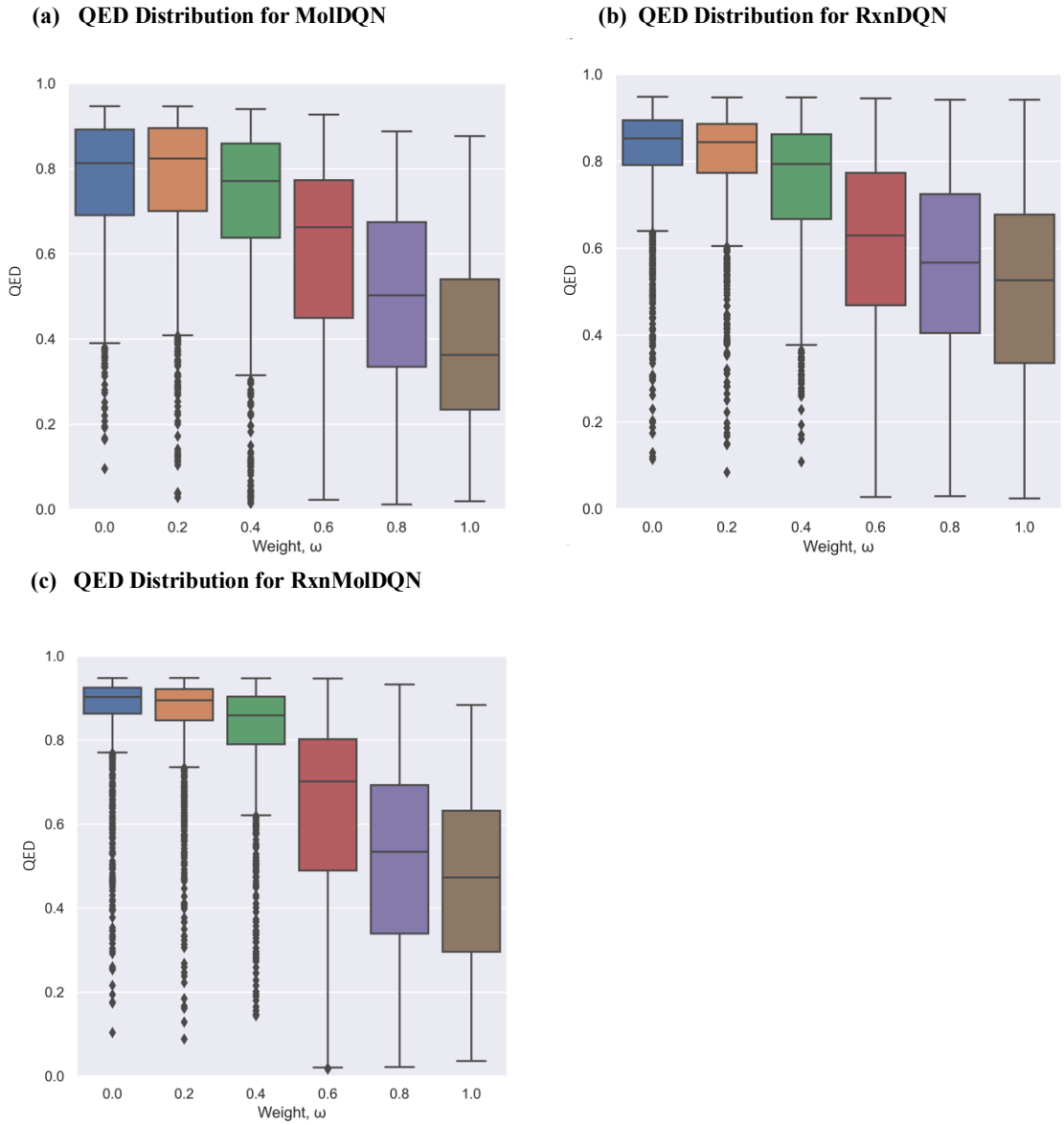


Figure 6.27: Distribution of QED values when performing multi-objective optimization between QED and Tanimoto similarity between the initial drug using Equation 2.41 with ω = 0, 0.2, 0.4, 0.6, 0.8, and 1.0: **(a)** MolDQN, **(b)** RxnDQN, and **(c)** RxnMolDQN. Each box plot was made by combining the QED score of the last 20 molecules generated when starting from each of the 100 DrugBank compounds (*i.e.,* each method consists of 2,000 QED scores to create the plot) for a particular objective weight. Reaction-based methods have better interquartile ranges of QED than the atom/bond-based method and thus generate more druglike molecules.

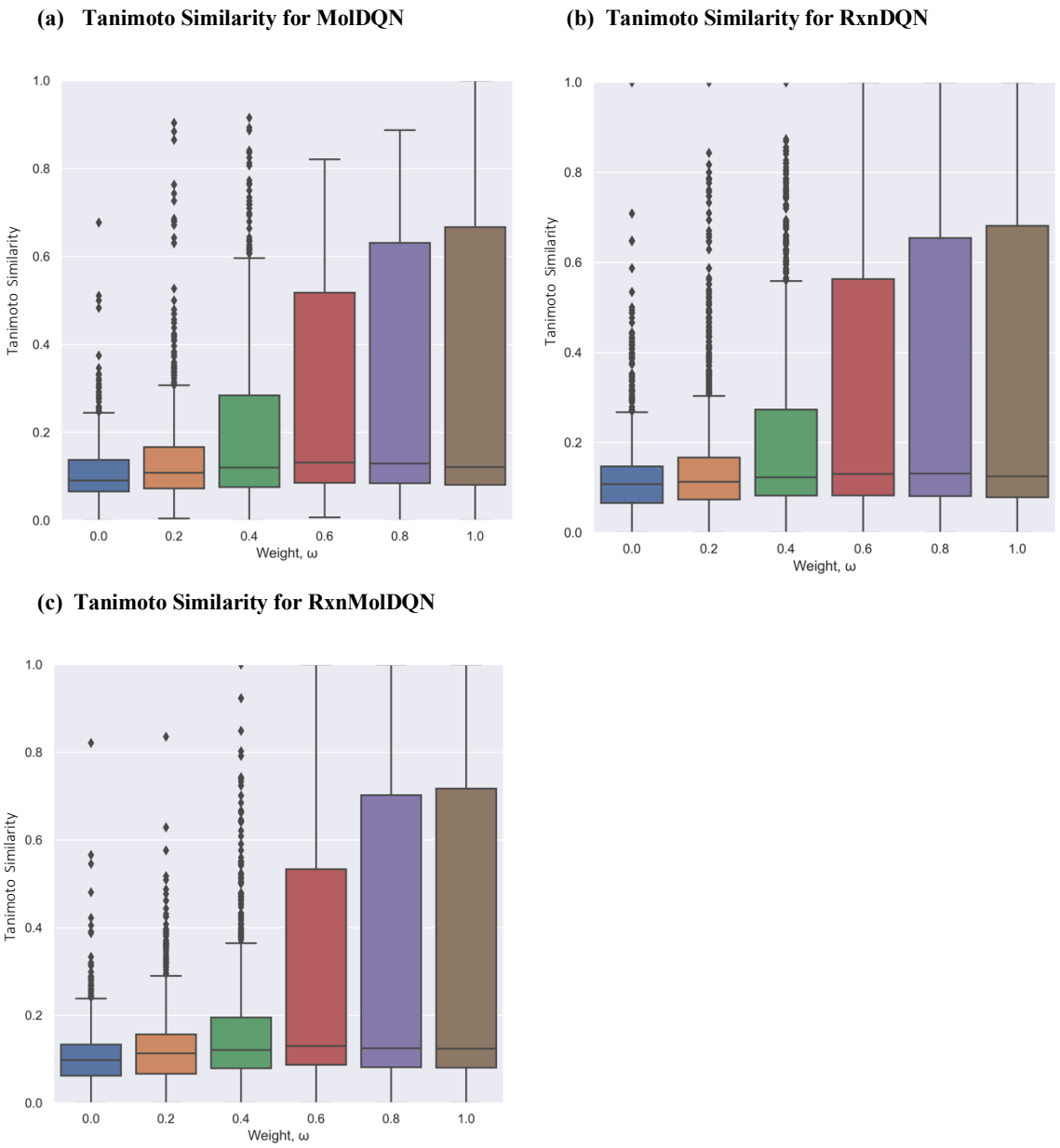


Figure 6.28: Distribution of Tanimoto similarity values when performing multi-objective optimization between QED and Tanimoto similarity between the initial drug using Equation 2.41 with ω = 0, 0.2, 0.4, 0.6, 0.8, and 1.0: **(a)** MolDQN, **(b)** RxnDQN, and **(c)** RxnMolDQN. Each box plot was made by combining the Tanimoto similarity score to the initial drug of the last 20 molecules generated when starting from each of the 100 DrugBank compounds (*i.e.,* each method consists of 2,000 Tanimoto similarity scores to create the plot) for a particular objective weight. There is no clear difference between the three methods in terms of Tanimoto similarity to the initial drug.

In the case of distribution for Tanimoto similarity in this multi-objective optimization (Figure 6.28), I expected to see an opposite trend as compared to QED: as the value of ω increases, the optimization of Tanimoto similarity improves. The expected trend was only observed across three methods up to $\omega = 0.4$ in Figures 6.28(a)-(c). This is not a surprising result, since we tend to only consider the top scored molecules for Tanimoto similarity unlike the sampling conducted in this scenario. By considering both Figures 6.27 and 6.28, all 2D-DQN methods have the capability to perform multi-objective optimization. However, the reactions-based methods perform better than MolDQN.

6.18.2 nc3D-DQN (LE) Methods

Figure 6.29 shows the reward progression for nc3D-DQN (LE) methods when performing the multi-objective optimization (Section 5.5.14) between predicted ligand efficiency and Tanimoto similarity to the initial starting ligand (Equation 3.14). All nc3D-DQN (LE) methods have the capability to perform multi-objective optimization between predicted ligand efficiency and Tanimoto similarity to Malhotra smaller ligands. When comparing the progression of multi-objective reward, nc3D-RxnMolDQN is the best optimizer followed by nc3D-RxnDQN and nc3D-MolDQN. As mentioned earlier in Section 6.14.2, the peak in predicted ligand efficiency observed in nc3D-MolDQN and nc3D-RxnMolDQN is caused by the tendency to make larger molecules over time (*i.e.*, more heavy atoms) after reaching the peak LE at around 150 episodes.

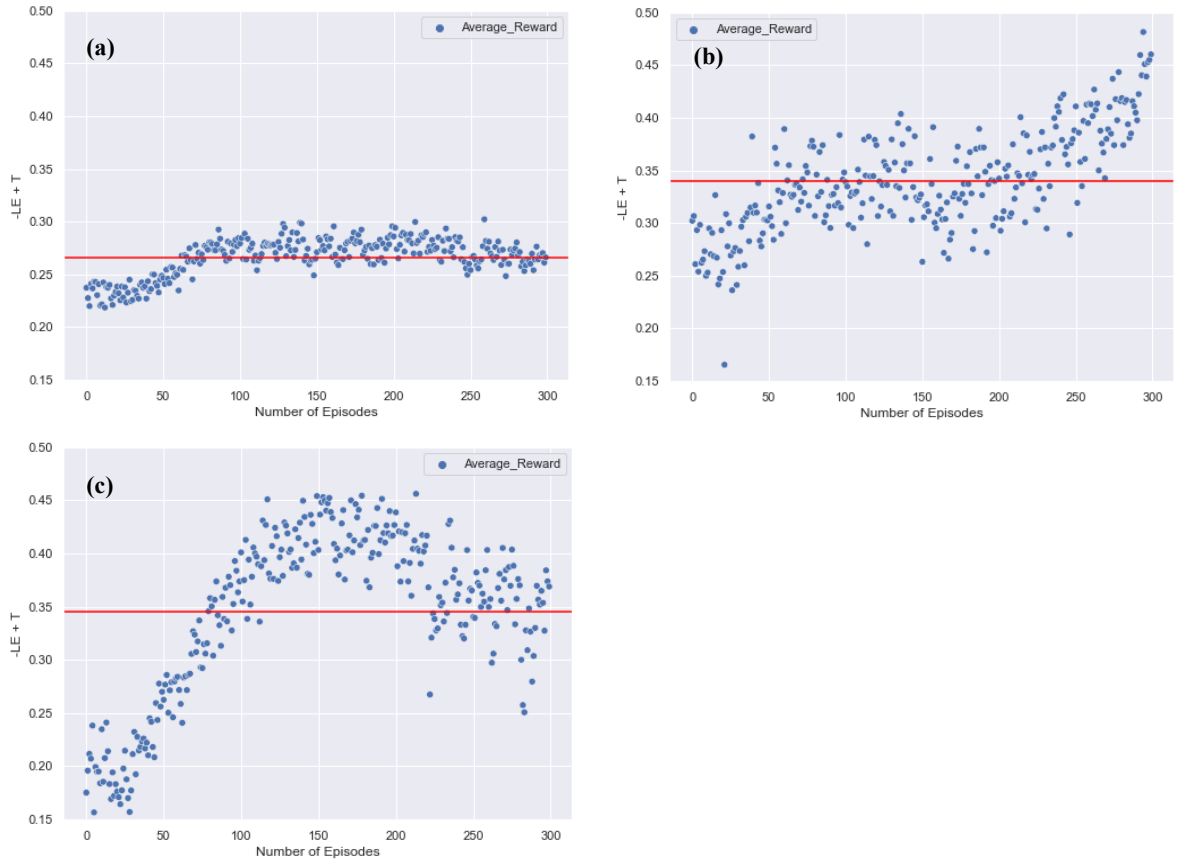


Figure 6.29: Reward progression for **(a)** nc3D-MolDQN, **(b)** nc3D-RxnDQN, and **(c)** nc3D-RxnMolDQN computed using average reward from 51 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) for each episode with the multi-objective reward function of LE and Tanimoto similarity (T) between smaller ligand where $\omega = 0.5$ (Equation 3.14). Red horizontal bars represent the average of reward values for each method (nc3D-MolDQN: 0.266; nc3D-RxnDQN: 0.340; nc3D-RxnMolDQN: 0.345). Progression of reward for the multi-objective optimization models is similar to the single-objective optimization models shown in Figure 6.24.

6.18.3 nc3D-DQN (nSuCOS) Methods

Figure 6.30 shows the reward progression for nc3D-DQN (nSuCOS) methods when performing the multi-objective optimization (Section 5.5.14) between normalized SuCOS and Tanimoto similarity to the Malhotra larger ligand (Equation 3.15). All nc3D-DQN (nSuCOS) methods were able to perform multi-objective optimization between normalized SuCOS and Tanimoto similarity between the generated molecules and larger ligands. When comparing the progression

of multi-objective reward, nc3D-MolDQN appears to be the best optimizer followed by nc3D-RxnMolDQN and nc3D-RxnDQN. As I noted in Section 6.14.3, this is likely because it is easier to optimize for normalized SuCOS when using smaller atom/bond-based actions than reaction-based actions.

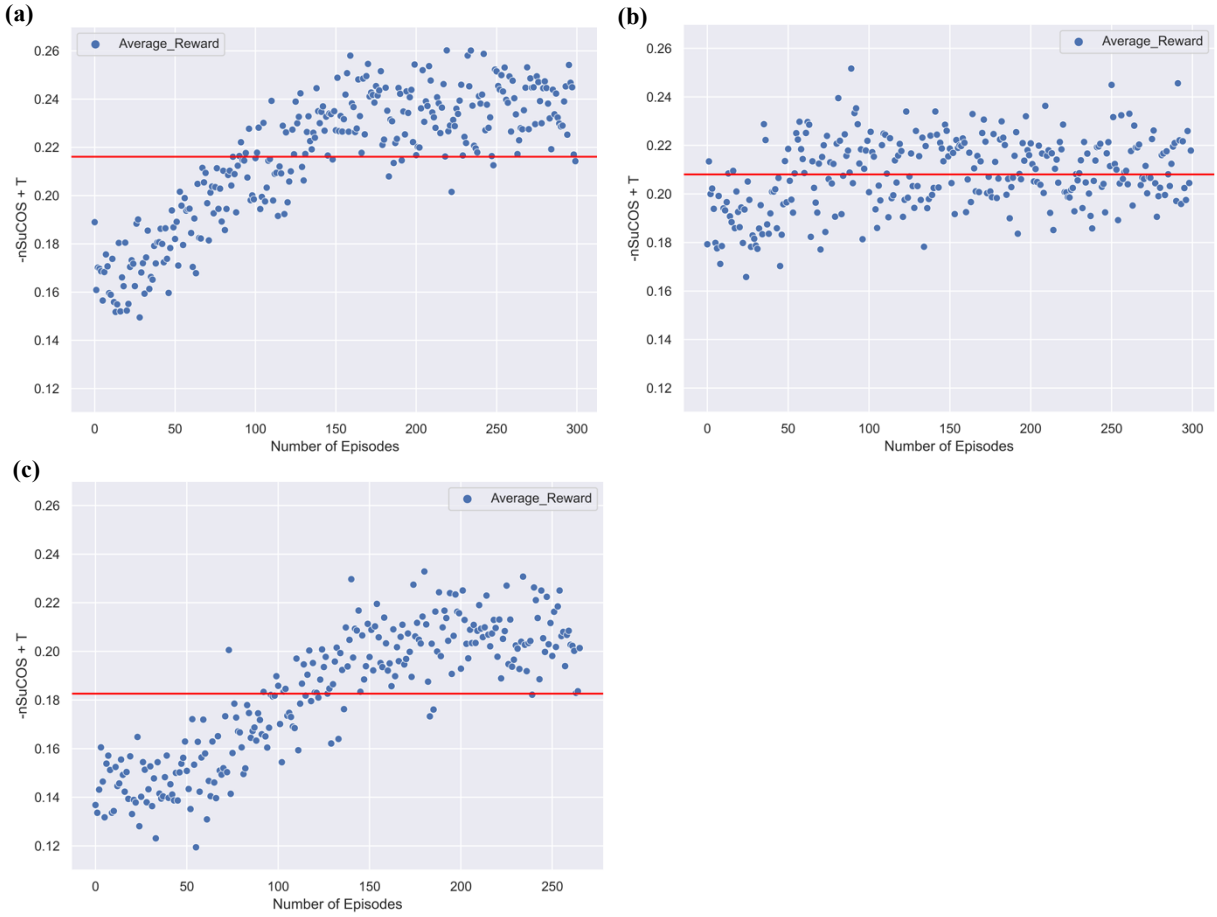


Figure 6.30: Reward progression for **(a)** nc3D-MolDQN, **(b)** nc3D-RxnDQN, and **(c)** nc3D-RxnMolDQN computed using average reward from 45 Malhotra smaller ligands (Malhotra & Karanicolas, 2017) for each episode with the multi-objective reward function of normalized SuCOS (nSuCOS) and Tanimoto similarity (T) between larger ligand where $\omega = 0.5$ (Equation 3.15). Red horizontal bars represent the average of reward values for each method (nc3D-MolDQN: 0.216; nc3D-RxnDQN: 0.208; nc3D-RxnMolDQN: 0.183). Progression of reward for the multi-objective optimization models is similar to the single-objective optimization models shown in Figure 6.25.

6.18.4 c3D-DQN Methods

Figure 6.31 shows the reward progression for c3D-DQN methods when performing the multi-objective optimization (Section 5.5.14) between predicted ligand efficiency and Tanimoto similarity to the initial covalent ligand (Equation 3.14). All three c3D-DQN methods were able to perform multi-objective optimization between predicted ligand efficiency and Tanimoto similarity between the desired covalent ligand and the generated molecule.

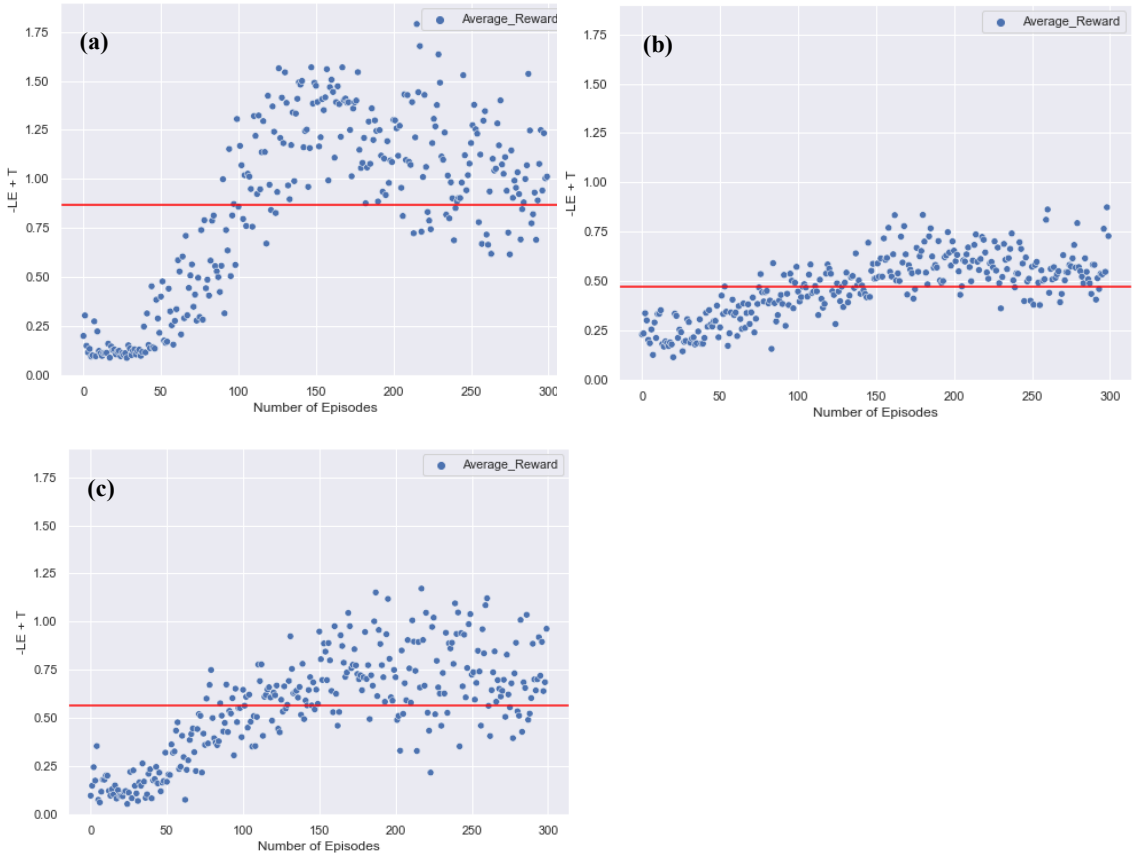


Figure 6.31: Reward progression for **(a)** c3D-MolDQN, **(b)** c3D-RxnDQN, and **(c)** c3D-RxnMolDQN computed using average reward from 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) for each episode with the multi-objective reward function of predicted ligand efficiency (LE) and Tanimoto similarity (T) between covalent ligand where $\omega = 0.5$ (Equation 3.14). Red horizontal bars represent the average of reward values for each method (c3D-MolDQN: 0.869; c3D-RxnDQN: 0.473; c3D-RxnMolDQN: 0.565). The progression of reward for the multi-objective optimization models is similar to the single-objective optimization models shown in Figure 6.26 expect for c3D-MolDQN.

When comparing the average reward (red horizontal bar), it is easier to perform the multi-objective optimization for c3D-MolDQN, followed by c3D-RxnMolDQN, and c3D-RxnDQN. This result is reasonable since it is easier for atom/bond-based action space to optimize for Tanimoto similarity especially in a constrained docking scenario where one part of the molecule needs to maintain the covalent warhead of Cys-145. Therefore, atom/bond-based action space in this case is rather more useful than the reaction-based action space to generate compounds with higher multi-objective rewards. The performance when performing multi-objective optimization improved in the order of c3D-RxnDQN, c3D-RxnMolDQN, and c3D-MolDQN.

6.19 Comparison of 2D-DQN against Other Published Methods

Tables 6.25 and 6.26 compare the performances in QED (Table 6.25) and penalized logP (Table 6.26) with its corresponding validity for the generated molecules (Tables 6.25-6.26) between 2D-DQN methods and other state-of-the-art techniques (Section 5.5.15). These techniques include ORGAN (Guimaraes *et al.*, 2017), JT-VAE (W. Jin *et al.*, 2018), and GCPN (You *et al.*, 2018). It has been found that across all metrics considered in Tables 6.25-6.26, RxnDQN and RxnMolDQN outperformed or performed as well as other state-of-the art techniques. Therefore, reaction-based methods present a promising alternative for *de novo* molecular design whilst optimizing multiple molecular properties noted in this chapter.

Table 6.25: Comparison of QED (top three best compounds with its corresponding averages and overall average across 3,000 episodes for each DrugBank compound) and validity for best models in 2D-DQN methods against literature methods using the reward function, QED. Literature methods include ORGAN, JT-VAE, and GCPN. Validity considers the percentage of compounds that are chemically valid according to the RDKit definition (Section 5.5.1). The values are approximated to three decimal places. Reactions-based RxnDQN and RxnMolDQN have state-of-the-art performance as compared to the literature methods.

Method	1st Best QED	Average of 1st Best QED	2nd Best QED	Average of 2nd Best QED	3rd Best QED	Average of 3rd Best QED	Overall Average QED	Validity (%)
RxnDQN	**0.948**	0.924±0.086	**0.948**	0.923±0.086	**0.948**	0.923±0.086	0.659±0.258	**100**
RxnMolDQN	**0.948**	**0.944±0.004**	**0.948**	**0.943±0.004**	**0.948**	**0.943±0.005**	**0.671±0.274**	**100**
MolDQN	0.947	0.849±0.125	0.947	0.844±0.126	0.947	0.841±0.126	0.418±0.275	**100**
ORGAN	0.896	-	0.824	-	0.820	-	-	2.2
JT-VAE	0.925	-	0.911	-	0.910	-	-	**100**
GCPN	**0.948**	-	0.947	-	0.946	-	-	**100**

Table 6.26: Comparison of penalized logP (top three best compounds with its corresponding averages and overall average across 3,000 episodes for each DrugBank compound) and validity for best models in 2D-DQN methods against literature methods using the reward function, penalized logP (p.logP). Literature methods include ORGAN, JT-VAE, and GCPN. Validity considers the percentage of compounds that are chemically valid according to the RDKit definition (Section 5.5.1). The values are approximated to three decimal places. RxnDQN and RxnMolDQN have state-of-the-art performance as compared to the literature methods.

Method	Best p.logP	Average of Best p.logP	2nd Best p.logP	Average of 2nd Best p.logP	3rd Best p.logP	Average of 3rd Best p.logP	Overall Average p.logP	Validity (%)
RxnDQN	**10.000**	9.420±2.528	9.999	9.415±2.527	**9.999**	9.412±2.526	5.951±4.768	**100**
RxnMolDQN	**10.000**	**9.995±0.005**	**10.000**	**9.990±0.007**	**9.999**	**9.986±0.009**	**6.207±4.747**	**100**
MolDQN	**10.000**	9.943±0.409	9.999	9.935±0.413	**9.999**	9.930±0.413	4.227±6.503	**100**
ORGAN	3.63	-	3.49	-	3.44	-	-	0.4
JT-VAE	5.30	-	4.93	-	4.49	-	-	**100**
GCPN	7.98	-	7.85	-	7.80	-	-	**100**

6.20 Conclusion and Future Directions

For 2D-DQN methods, my new reinforcement learning *de novo* drug design methods of RxnDQN and RxnMolDQN incorporate novel reaction-based action spaces and adopt a pragmatic approach to computational drug design by exploiting commercially available reagents extracted from high

yield reactions in Reaxys. My reaction-based 2D-DQN methods are completely extensible and allow the introduction of new user-defined reactions. Both reaction-based methods outperform the atom/bond-based MolDQN (Zhou *et al.*, 2019) in terms of reward optimization (Sections 6.4-6.5), multi-objective optimization (Section 6.18.1), Walters' compound quality (Section 6.8), silly walks score (Section 6.13), synthetic accessibility (Section 6.3), computational efficiency (Section 6.16), Tanimoto similarity to the target drug (Section 6.10.2), and chemical space coverage (Section 6.11). In other words, my reaction-based methods design molecules that are more synthetically accessible whilst being 100% chemically valid (Section 6.2) and improving various molecular properties important to drug discovery. The hybrid action space of RxnMolDQN is the best method since it enables the greatest exploration of chemical space in addition to optimizing molecular properties.

Although the project aim (Section 2.2.14) has been achieved for 2D-DQN methods, there are other factors that I could potentially explore such as the impact of adding additional reactions other than the top 10 reactions (Roughley and Jordan, 2011), using reagents other than the Reaxys database (Elsevier, 2020), and exploring alternative multi-objective reward functions such as incorporating elements directly from the Lipinski's rule of five (Section 5.5.13) (Lipinski *et al.*, 2001). Even though these 2D-DQN methods can generate compounds that are 100% chemically valid, the validity defined by RDKit did not lead to zero reactive or unstable functional groups especially for MolDQN (see Figure 6.8 for examples). Therefore, this is also another area of exploration where I could potentially include a quantum mechanics electrophilicity prediction to remove reactive functional groups or use a more formulated "silliness" filter to remove unprecedented or unstable functional groups.

For nc3D-DQN (LE) methods, my new structure-based reinforcement learning method incorporates non-constrained docking using AutoDock Vina and has extended the *de novo* drug design methods introduced in 2D-DQN methods. Like the 2D-DQN methods, reaction-based nc3D-DQN (LE) methods are completely extensible and allow user-defined reactions. Both reaction-based methods outperform atom/bond-based nc3D-MolDQN in terms of reward optimization (Section 6.6.1), synthetic accessibility (Section 6.3), Walters' compound quality (Section 6.8), silly walks score (Section 6.13), computational efficiency (Section 6.16), and Lipinski's rule of five (Section 6.17). The results for rediscovery (Section 6.9.2) suggested that despite the least chemical space coverage reported in Section 6.11, nc3D-RxnDQN performed the best at rediscovering/generating both smaller and larger Malhotra ligands. This suggests that focusing only on chemical space coverage does not directly correlate with whether a method is exploring the region of chemical space that is considered useful in the context of nc3D-DQN (LE) methods. It was also found that it is likely to be easier for atom/bond-based nc3D-MolDQN to retain the shape and feature overlaps defined by normalized SuCOS (Section 6.12), since this method can make the least chemical modifications to a molecule. For Tanimoto similarity results (Section 6.10.2), the nc3D-DQN (LE) methods generated molecules that are more structurally similar to the starting molecule (*i.e.*, smaller ligand) than the target molecule (*i.e.*, larger ligand). Therefore, like the 2D-DQN methods, the reaction-based nc3D-DQN methods propose molecules that are more synthetically accessible whilst generating molecules that are 100% chemically valid (Section 6.2) and improving various molecular properties important to drug discovery. Although the hybrid action space of nc3D-RxnMolDQN explored more chemical space than nc3D-RxnDQN and acted as a better optimizer in both single and multi-objective reward contexts, the best method for nc3D-DQN methods is nc3D-RxnDQN. This is particularly because of the performance reported in rediscovery.

The work for nc3D-DQN (LE) methods has reached its project aim (Section 3.2.8), but it may be interesting to incorporate group efficiency and fit quality as a single optimization reward function for nc3D-DQN (LE) methods (Schultes *et al.*, 2010). This is because ligand efficiency could cause values to fluctuate notably for smaller compounds and the scale depends on the size of the starting fragment, which affects its contribution to a reward function when combined in a simple weighting with other objectives. It is also worth noting that calculating ligand efficiency in this manner can potentially be a limiting factor when performing iterative growth where the addition of linker atoms can cause ligand efficiency to drop before a higher binding affinity interaction is reached and ligand efficiency increases.

For nc3D-DQN (nSuCOS) methods, I have introduced a new pharmacophoric-based reward function to the nc3D-DQN (LE) methods by using normalized SuCOS overlap with a desired active ligand. Both reaction-based methods outperform atom/bond-based nc3D-MolDQN in terms of synthetic accessibility (Section 6.3), Walters' compound quality (Section 6.8), and silly walks score (Section 6.13). Reaction-based nc3D-RxnDQN outperformed the other two methods in terms of computational efficiency (Section 6.16), Tanimoto similarity (Section 6.10.2) and Lipinski's rule of five (Section 6.17). nc3D-RxnMolDQN was found to have the greatest chemical space coverage (Section 6.11) followed by nc3D-MolDQN and nc3D-RxnDQN. It was also found that it is difficult for the pharmacophoric-based reward function to achieve rediscovery (Section 6.9.3) than the predicted ligand efficiency (Section 6.9.2) when considering the differences in receptors PDBQT preparation (Section 5.4.4) and rediscovery of Malhotra larger ligands. In terms of the reward optimization for normalized SuCOS (Section 6.7), the performances between nc3D-DQN methods were similar. Even though nc3D-MolDQN performed better in terms of optimization efficiency for both single and multi-objectives (Sections 6.14.3 and 6.18.3), this is

likely because the atom/bond-based action space can make smaller chemical modifications than the reaction-based action space, so it is easier to retain the shape and feature overlaps defined by normalized SuCOS, as discussed in Section 6.7. Although nc3D-MolDQN was found to be a better optimizer, by considering other metrics important to drug discovery in this chapter, reaction-based methods performed better than nc3D-MolDQN for the reward function of normalized SuCOS. In terms of the best method for normalized SuCOS, this would be nc3D-RxnMolDQN due to the performance in both chemical space coverage computed using UMAP and reward optimization.

It was also intriguing to find that the usage of normalized SuCOS as the reward function avoids the duplication of molecules (Section 6.15) to the same extent as the reward function of QED. The nc3D-DQN methods for both predicted ligand efficiency and normalized SuCOS performed similarly on Tanimoto similarity between the smaller and larger ligands (Section 6.10.2), but the predicted ligand efficiency-based models were better at rediscovering larger ligands (Sections 6.9.2-6.9.3). This may suggest that the reward function of normalized SuCOS is exploring a chemical space that is different to that of the predicted ligand efficiency. For future work, it may be worthwhile to explore the chemical space when optimizing both predicted ligand efficiency and normalized SuCOS together in multi-objective optimization. This can potentially further guide the exploration of chemical space towards the desired active molecule while also binding efficiently. Moreover, when considering the percentage of ligands that satisfied the normalized SuCOS threshold (Section 6.12), the reward function of predicted ligand efficiency performed better than the reward function of normalized SuCOS in two out of the three nc3D-DQN methods. This likely suggests the difficulty in optimizing for normalized SuCOS in the single-objective optimization.

The work for nc3D-DQN (nSuCOS) methods has reached its project aim (Section 3.2.8), but I would like to explore how the algorithm can be biased towards the generation of target-specific molecules such as kinase inhibitors, G-Protein Coupled Receptor (GPCR) antagonists, or helicase inhibitors.

For c3D-DQN methods, I have introduced new structure-based reinforcement learning methods that incorporate constrained docking using AutoDock Vina that can be used for both covalent ligand design, and in fragment-based drug design. Like the nc3D-DQN methods, reaction-based c3D-DQN methods are completely extensible and allow user-defined reactions. I have presented a case study using SARS-CoV-2 main protease in this chapter, but this methodology is applicable to other targets, as explained in Section 4.3.1. Both reaction-based c3D-RxnDQN and c3D-RxnMolDQN outperform atom/bond-based c3D-MolDQN in terms of synthetic accessibility (Section 6.3), Walters' compound quality (Section 6.8), silly walks (Section 6.13), rediscovery (Section 6.9.4), and computational efficiency (Section 6.16). I found that when optimizing for predicted ligand efficiency with the c3D-DQN methods, it was easier for the atom/bond-based method to perform optimization than the pure reaction-based c3D-RxnDQN, due to the restrictions on chemical modifications. This observation was also consistent with the chemical space coverage results (Section 6.11), where c3D-RxnMolDQN demonstrated the greatest exploration followed by c3D-MolDQN and c3D-RxnDQN. However, atom/bond-based c3D-MolDQN was found to be a better optimizer than the reaction-based methods (Section 6.14.4). It is also interesting to note that the c3D-DQN methods can generate molecules that are more structurally like the desired covalent ligands as compared to nc3D-DQN methods (Section 6.10.2), but it proved rather difficult for them to retain the shape and feature overlaps defined by normalized SuCOS (Section 6.12). In terms of multi-objective reward (Section 6.18.4) and

Lipinski's rule of five (Section 6.17), c3D-MolDQN performed better than the reaction-based methods. This is likely due to the nature of atom/bond-based action space enabling smaller chemical modifications, which makes it easier to optimize for Tanimoto similarity and also to retain the desired pharmacophoric features. Therefore, overall, like the nc3D-DQN methods, the reaction-based c3D-DQN methods propose molecules that are both synthetically accessible and chemically valid (Section 6.2), whilst simultaneously optimizing for other molecular properties important to drug discovery. For c3D-DQN methods, the best method is c3D-RxnMolDQN given its largest exploration in chemical space whilst retaining rediscovery and the advantages of c3D-RxnDQN.

It would be interesting to extend the scope of the work shown in this chapter to incorporate all ligands relevant to SARS-CoV-2 main protease and perform a thorough study to propose new potential inhibitors with wet-lab assays. In addition, the current c3D-DQN methods consider the constrained docking by recognizing and covalently modifying the residue within a target protein. However, as explained in Section 4.2.1, there is another type of constrained docking where we keep fixed a known bound fragment or substructure of a non-covalent ligand. Therefore, if a future study could be made, I would like to explore this type of constrained docking as well.

7 Conclusion and Future Directions

I have developed seven novel deep Q-learning-based *de novo* drug design methods (Figure 1.17). These include ligand-based 2D-DQN (reaction-based RxnDQN and hybrid reaction/atom-bond based RxnMolDQN); and structure-based 3D-DQN methods (non-constrained nc3D-DQN and constrained c3D-DQN methods). When considering the reward functions, there are 22 different methods in total (*i.e.*, reward with QED, predicted ligand efficiency, and normalized SuCOS for single-objectives and single-objectives combined with Tanimoto similarity to a desired or known active ligand for multi-objectives).

In Chapter 2, I introduced novel reaction-based action spaces by incorporating the top 10 most commonly used reactions in medicinal chemistry (Roughley and Jordan, 2011) together with reagents from Reaxys. The reaction-based action space led to the development of RxnDQN. By hybridizing both reaction and atom/bond-based action spaces, RxnMolDQN demonstrated an ability to explore chemical space both "locally" and "globally". The results in Chapter 6 showed that my reaction-based methods outperformed the previously developed atom/bond-based action space of MolDQN (Zhou *et al.*, 2019) in metrics such as chemical space coverage and synthetic accessibility. My reaction-based RxnDQN and RxnMolDQN were also found to perform as well or better than state-of-the-art techniques available in the literature as noted in Section 6.19. Despite these advantages, a major limitation of ligand-based *de novo* methods is that they ignore the existence of the binding pocket.

Therefore, in Chapter 3, I developed a new non-constrained structure-based *de novo* algorithms that model how ligands bind to the active site of a protein, using the open-source docking toolkits, AutoDockTools (G. M. Morris *et al.*, 2009) and AutoDock Vina (Trott and Olson, 2010). As

shown in Chapter 6, nc3D-DQN (LE) methods led to the generation of new molecules that are capable of binding to the active site of a protein by proposing drug-like molecules as compared to the previous 2D-DQN methods. In addition, the reaction-based nc3D-DQN (LE) methods of nc3D-RxnDQN and nc3D-RxnMolDQN improved on metrics such as synthetic accessibility, reward optimization, rediscovery, and Lipinski's rule of five as compared to the atom/bond-based nc3D-MolDQN. It was also noted that for the nc3D-DQN (LE) methods, there is a difference between the overall chemical space coverage and chemical space coverage that is considered useful (Section 6.11). Similar to 2D-DQN methods, the reaction-based nc3D-DQN (LE) methods outperformed the atom/bond-based nc3D-MolDQN.

In Chapter 3, I also introduced a new pharmacophoric-based reward function to nc3D-DQN (LE) methods by using normalized Combined Overlap Score with a known, desired active compound (SuCOS). As shown in Chapter 6, the reaction-based nc3D-DQN (nSuCOS) methods outperformed the atom/bond-based nc3D-MolDQN in metrics such as synthetic accessibility, Walters' compound quality, and silly walks score (Sections 6.3, 6.8, and 6.13). But it was interesting to find that the reward function of normalized SuCOS is harder to optimize than predicted ligand efficiency. In addition, normalized SuCOS reward function avoids the generation of the duplicate molecules to the same extent as when using the reward function of QED.

In order to further extend the capability to perform docking using AutoDock Vina, I developed a new constrained structure-based *de novo* algorithms in Chapter 4. As described in Chapter 6, this resulted in the generation of molecules that are capable of binding to the active site of a protein while having greater confidence in the predicted binding mode. The reaction-based c3D-DQN methods outperformed atom/bond-based c3D-MolDQN in metrics such as synthetic accessibility,

rediscovery, and computational efficiency (Sections 6.3, 6.9, and 6.16). In a constrained docking scenario, it was easier to optimize using atom/bond-based methods than the reaction-based method, since there are more restrictions on the possible small chemical modifications imposed by the covalent warhead for the reaction-based action space. It was also interesting to note that in general, c3D-DQN methods generated structurally similar molecules to the starting ligands with a better average Tanimoto similarity score than nc3D-DQN methods, but it was rather difficult to retain the shape and feature overlaps defined by normalized SuCOS (Leung *et al.*, 2019).

Therefore, by exploring a combination of novel deep Q-learning-based *de novo* drug design methods and a variety of reward functions, I have shown that reaction-based methods outperform atom/bond-based methods when optimizing for desirable molecular properties in drug discovery.

Although I have achieved the overall goal of this project by developing and testing novel *de novo* design algorithms, in terms of future directions mentioned in Section 6.20, it would be interesting to explore multi-objective reward functions further with these methods. This is because in real-world drug discovery, we need to design compounds that are also affordable, non-toxic, bioavailable, metabolically stable, in addition to being able to bind to the target (or targets) of interest. Therefore, it would be exciting to explore more sophisticated multi-objective reward functions that combine 2D and 3D properties, for example: predicted compound price (using CoPriNet (Sanchez-Garcia *et al.*, 2023)) plus predicted ligand efficiency plus predicted toxicity (J. Zhang *et al.*, 2021).

The code and data from this project will be made available on GitHub (https://github.com/angoto) when I submit the preprints, which is currently under preparation.

References

Abad-Zapatero, C. and Metz, J. T. Ligand efficiency indices as guideposts for drug discovery. *Drug Discov Today*. **2005**, 10(7), 464-469.

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M. *et al.* (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/ (accessed December 5, 2019).

Agarap, A. F. Deep Learning using Rectified Linear Units (ReLU). *arXiv*. **2018**, DOI: 10.48550/ARXIV.41803.08375.

Aldeghi, M., Heifetz, A., Bodkin, M. J., Knapp, S. and Biggin, P. C. Accurate calculation of the absolute free energy of binding for drug molecules. *Chem Sci*. **2016**, 7(1), 207-218.

Allinger, N. L. Conformational analysis. 130. MM2. A hydrocarbon force field utilizing V1 and V2 torsional terms. *J. Am. Chem. Soc.* **1977**, 99(25), 8127–8134.

Amaravadhi, H., Baek, K. and Yoon, H. S. Revisiting de novo drug design: receptor based pharmacophore screening. *Curr Top Med Chem*. **2014**, 14(16), 1890-1898.

Arjovsky, M., Chintala, S. and Bottou, L. Wasserstein GAN. *arXiv*. **2017**, DOI: 10.48550/arXiv.41701.07875.

Ashwini, M., Murugan, S. B., Balamurugan, S. and Sathishkumar, R. [Advances in Molecular Cloning]. *Mol Biol (Mosk)*. **2016**, 50(1), 3-9.

Aykul, S. and Martinez-Hackert, E. Determination of half-maximal inhibitory concentration using biosensor-based protein interaction analysis. *Anal Biochem*. **2016**, 508, 97-103.

Badger, M. R. and Price, G. D. The Role of Carbonic Anhydrase in Photosynthesis. *Annu. rev. plant physiol. plant mol. biol.* **1994**, 45(1), 369-392.

Ballester, P. J. and Mitchell, J. B. A machine learning approach to predicting protein-ligand binding affinity with applications to molecular docking. *Bioinformatics*. **2010**, 26(9), 1169-1175.

Bento, A. P., Gaulton, A., Hersey, A., Bellis, L. J., Chambers, J., Davies, M., Krüger, F. A., Light, Y., Mak, L., McGlinchey, S. *et al.* The ChEMBL bioactivity database: an update. *Nucleic Acids Res*. **2014**, 42(Database issue), D1083-1090.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. and Bourne, P. E. The Protein Data Bank. *Nucleic Acids Res*. **2000**, 28(1), 235-242.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kotter, T., Meinl, T., Ohl, P., Thiel, K. and Wiswedel, B. KNIME - the Konstanz information miner: version 2.0 and beyond. *SIGKDD Explor. Newsl.* **2009**, 11(1), 26-31.

Bertsekas, D. P. *Dynamic Programming and Optimal Control, Vol. II*; Athena Scientific, 2007.

Bian, Y. and Xie, X. Q. Generative chemistry: drug discovery with deep learning generative models. *J Mol Model*. **2021**, 27(3), 71.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S. and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nat Chem*. **2012**, 4(2), 90-98.

Blaschke, T., Arús-Pous, J., Chen, H., Margreitter, C., Tyrchan, C., Engkvist, O., Papadopoulos, K. and Patronov, A. REINVENT 2.0: An AI Tool for De Novo Drug Design. *J Chem Inf Model*. **2020**, 60(12), 5918-5922.

Blaschke, T., Olivecrona, M., Engkvist, O., Bajorath, J. and Chen, H. Application of Generative Autoencoder in De Novo Molecular Design. *Mol Inform*. **2018**, 37(1-2).

Bo, G. Giuseppe Brotzu and the discovery of cephalosporins. *Clin Microbiol Infect*. **2000**, 6 Suppl 3, 6-9.

Bohacek, R., Mcmartin, C., Glunz, P. and Rich, D. H. (1999). Growmol, A De novo Computer Program, and its Application to Thermolysin and Pepsin: Results of the Design and Synthesis of a Novel Inhibitor. In D. G. Truhlar, W. J. Howe, A. J. Hopfinger, J. Blaney and R. A. Dammkoehler (Eds.), Rational Drug Design (pp. 103-114). New York, NY: Springer New York.

Böhm, H. J. The computer program LUDI: a new method for the de novo design of enzyme inhibitors. *J Comput Aided Mol Des*. **1992**, 6(1), 61-78.

Boitreaud, J., Mallet, V., Oliver, C. and Waldispühl, J. OptiMol: Optimization of Binding Affinities in Chemical Space for Drug Discovery. *J Chem Inf Model*. **2020**, 60(12), 5658-5666.

Bosch, F. and Rosich, L. The contributions of Paul Ehrlich to pharmacology: a tribute on the occasion of the centenary of his Nobel Prize. *Pharmacology*. **2008**, 82(3), 171-179.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge, UK; Cambridge University Press, 2004.

Boyles, F., Deane, C. M. and Morris, G. M. Learning from the ligand: using ligand-based features to improve binding affinity prediction. *Bioinformatics*. **2020**, 36(3), 758-764.

Breiman, L. Random Forests. *Mach. Learn.* **2001**, 45(1), 5-32.

Brooks, E. A., Galarza, S., Gencoglu, M. F., Cornelison, R. C., Munson, J. M. and Peyton, S. R. Applicability of drug response metrics for cancer studies using biomaterials. *Philos Trans R Soc Lond B Biol Sci*. **2019**, 374(1779), 20180226.

Brown, B. P., Mendenhall, J., Geanes, A. R. and Meiler, J. General Purpose Structure-Based Drug Discovery Neural Network Score Functions with Human-Interpretable Pharmacophore Maps. *J Chem Inf Model*. **2021**, 61(2), 603-620.

Brown, N., Fiscato, M., Segler, M. H. S. and Vaucher, A. C. GuacaMol: Benchmarking Models for de Novo Molecular Design. *J Chem Inf Model*. **2019**, 59(3), 1096-1108.

Burley, S. K. Impact of structural biologists and the Protein Data Bank on small-molecule drug discovery and development. *J Biol Chem*. **2021**, 296, 100559.

Chan, H. C. S., Shan, H., Dahoun, T., Vogel, H. and Yuan, S. Advancing Drug Discovery via Artificial Intelligence. *Trends Pharmacol Sci*. **2019**, 40(8), 592-604.

Chan, H. T. H., Moesser, M. A., Walters, R. K., Malla, T. R., Twidale, R. M., John, T., Deeks, H. M., Johnston-Wood, T., Mikhailov, V., Sessions, R. B. *et al.* Discovery of SARS-CoV-2 M(pro) peptide inhibitors from modelling substrate and ligand binding. *Chem Sci*. **2021**, 12(41), 13686-13703.

Chen, H., Engkvist, O., Wang, Y., Olivecrona, M. and Blaschke, T. The rise of deep learning in drug discovery. *Drug Discov Today*. **2018**, 23(6), 1241-1250.

Cherkasov, A., Muratov, E. N., Fourches, D., Varnek, A., Baskin, II, Cronin, M., Dearden, J., Gramatica, P., Martin, Y. C., Todeschini, R. *et al.* QSAR modeling: where have you been? Where are you going to? *J Med Chem*. **2014**, 57(12), 4977-5010.

Clark, D. E., Frenkel, D., Levy, S. A., Li, J., Murray, C. W., Robson, B., Waszkowycz, B. and Westhead, D. R. PRO-LIGAND: an approach to de novo molecular design. 1. Application to the design of organic molecules. *J Comput Aided Mol Des*. **1995**, 9(1), 13-32.

Clark, M., Guarnieri, F., Shkurko, I. and Wiseman, J. Grand canonical Monte Carlo simulation of ligand-protein binding. *J Chem Inf Model*. **2006**, 46(1), 231-242.

Cortes, C. and Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, 20(3), 273-297.

Cox, O. B., Krojer, T., Collins, P., Monteiro, O., Talon, R., Bradley, A., Fedorov, O., Amin, J., Marsden, B. D., Spencer, J. *et al.* A poised fragment library enables rapid synthetic expansion yielding the first reported inhibitors of PHIP(2), an atypical bromodomain. *Chem Sci*. **2016**, 7(3), 2322-2330.

Cramer, R. D., Patterson, D. E. and Bunce, J. D. Comparative molecular field analysis (CoMFA). 1. Effect of shape on binding of steroids to carrier proteins. *J Am Chem Soc*. **1988**, 110(18), 5959-5967.

Dalke, A., Hert, J. and Kramer, C. mmpdb: An Open-Source Matched Molecular Pair Platform for Large Multiproperty Data Sets. *J Chem Inf Model*. **2018**, 58(5), 902-910.

Danziger, D. J. and Dean, P. M. Automated site-directed drug design: a general algorithm for knowledge acquisition about hydrogen-bonding regions at protein surfaces. *Proc R Soc Lond B Biol Sci*. **1989**, 236(1283), 101-113.

Daylight-Chemical-Information-Systems (1997). 4. SMARTS - A Language for Describing Molecular Patterns. https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (accessed September 27, 2022).

De Cao, N. and Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv*. **2018**, DOI: 10.48550/arXiv.41805.11973.

Del Re, G. 812. A simple MO–LCAO method for the calculation of charge distributions in saturated organic molecules. *J. Chem. Soc. (resumed)*. **1958**, (0), 4031-4040.

Devi, R. V., Sathya, S. S. and Coumar, M. S. Evolutionary algorithms for de novo drug design - A survey. *J Appl. Soft Comput.* **2015**, 27(C), 543-552.

DiMasi, J. A., Grabowski, H. G. and Hansen, R. W. Innovation in the pharmaceutical industry: New estimates of R&D costs. *J Health Econ*. **2016**, 47, 20-33.

Douangamath, A., Fearon, D., Gehrtz, P., Krojer, T., Lukacik, P., Owen, C. D., Resnick, E., Strain-Damerell, C., Aimon, A., Ábrányi-Balogh, P. *et al.* Crystallographic and electrophilic fragment screening of the SARS-CoV-2 main protease. *Nat Commun*. **2020**, 11(1), 5047.

Drew, K. L., Baiman, H., Khwaounjoo, P., Yu, B. and Reynisson, J. Size estimation of chemical space: how big is it? *J Pharm Pharmacol*. **2012**, 64(4), 490-495.

Drews, J. Drug discovery: a historical perspective. *Science*. **2000**, 287(5460), 1960-1964.

Durrant, J. D. and McCammon, J. A. NNScore 2.0: a neural-network receptor-ligand scoring function. *J Chem Inf Model*. **2011**, 51(11), 2897-2903.

Edwards, A. M., Arrowsmith, C. H., Christendat, D., Dharamsi, A., Friesen, J. D., Greenblatt, J. F. and Vedadi, M. Protein production: feeding the crystallographers and NMR spectroscopists. *Nat Struct Biol*. **2000**, 7 Suppl, 970-972.

Eldridge, M. D., Murray, C. W., Auton, T. R., Paolini, G. V. and Mee, R. P. Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. *Journal of Computer-Aided Molecular Design*. **1997**, 11(5), 425-445.

Elsevier (2020). Reaxys. https://www.reaxys.com (accessed December 5, 2020).

Elton, D. C., Boukouvalas, Z., Fuge, M. D. and Chung, P. W. Deep learning for molecular design—a review of the state of the art. *Mol. Syst. Des. Eng.* **2019**, 4(4), 828-849.

Erlanson, D. A., Fesik, S. W., Hubbard, R. E., Jahnke, W. and Jhoti, H. Twenty years on: the impact of fragments on drug discovery. *Nat Rev Drug Discov*. **2016**, 15(9), 605-619.

Ertl, P. and Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J Cheminform*. **2009**, 1(1), 8.

Fausett, L. V. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*; Prentice-Hall, 1994. https://dl.acm.org/doi/book/10.5555/197023

FDA (2018). Step 3: Clinical Research. https://www.fda.gov/patients/drug-development-process/step-3-clinical-research (accessed November 5, 2022).

FDA (2021). Fact Sheet: FDA at a Glance. https://www.fda.gov/about-fda/fda-basics/fact-sheet-fda-glance#:~:text=FDA%2DRegulated%20Products%20and%20Facilities&text=This%20includes%20everything%20we%20eat,FDA%2Dapproved%20animal%20drug%20products. (accessed November 5, 2022).

Fitzner, M. Roche, Basel, Switzerland. Personal Communication, 2019.

Fitzner, M. Roche, Basel, Switzerland. Personal Communication, 2020.

Fleming, A. On the Antibacterial Action of Cultures of a Penicillium, with Special Reference to their Use in the Isolation of B. influenzæ. *Br J Exp Pathol*. **1929**, 10(3), 226-236.

Friesner, R. A., Banks, J. L., Murphy, R. B., Halgren, T. A., Klicic, J. J., Mainz, D. T., Repasky, M. P., Knoll, E. H., Shelley, M., Perry, J. K. *et al.* Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy. *J Med Chem*. **2004**, 47(7), 1739-1749.

Gabel, J., Desaphy, J. and Rognan, D. Beware of machine learning-based scoring functions-on the danger of developing black boxes. *J Chem Inf Model*. **2014**, 54(10), 2807-2815.

García, I., Fall, Y. and Gómez, G. Review of QSAR for DNA polymerase inhibitors and new models for heterogeneous series of compounds. *Curr Comput Aided Drug Des*. **2011**, 7(4), 249-254.

Gentile, F., Agrawal, V., Hsing, M., Ton, A. T., Ban, F., Norinder, U., Gleave, M. E. and Cherkasov, A. Deep Docking: A Deep Learning Platform for Augmentation of Structure Based Drug Discovery. *ACS Cent Sci*. **2020**, 6(6), 939-949.

Gershell, L. J. and Atkins, J. H. A brief history of novel drug discovery technologies. *Nat Rev Drug Discov*. **2003**, 2(4), 321-327.

Gillet, V. J., Myatt, G., Zsoldos, Z. and Johnson, A. P. SPROUT, HIPPO and CAESA: Tools for de novo structure generation and estimation of synthetic accessibility. *Perspectives in Drug Discovery and Design*. **1995**, 3(1), 34-50.

Gillet, V. J., Newell, W., Mata, P., Myatt, G., Sike, S., Zsoldos, Z. and Johnson, A. P. SPROUT: recent developments in the de novo design of molecules. *J Chem Inf Comput Sci*. **1994**, 34(1), 207-217.

Gilson, M. K. and Zhou, H. X. Calculation of protein-ligand binding affinities. *Annu Rev Biophys Biomol Struct*. **2007**, 36, 21-42.

Glover, F. Tabu Search—Part I. *ORSA J. Comput.* **1989**, 1(3), 190-206.

Gohlke, H., Hendlich, M. and Klebe, G. Knowledge-based scoring function to predict protein-ligand interactions. *J Mol Biol*. **2000**, 295(2), 337-356.

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. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent Sci*. **2018**, 4(2), 268-276.

Goodford, P. J. Drug design by the method of receptor fit. *J Med Chem*. **1984**, 27(5), 558-564.

Goodford, P. J. A computational procedure for determining energetically favorable binding sites on biologically important macromolecules. *J Med Chem*. **1985**, 28(7), 849-857.

Goodman, J. M., Pletnev, I., Thiessen, P., Bolton, E. and Heller, S. R. InChI version 1.06: now more than 99.99% reliable. *J Cheminform*. **2021**, 13(1), 40.

Graff, D. E., Shakhnovich, E. I. and Coley, C. W. Accelerating high-throughput virtual screening through molecular pool-based active learning. *Chem Sci*. **2021**, 12(22), 7866-7881.

Gritsch, S., Guccione, S., Hoffmann, R., Cambria, A., Raciti, G. and Langer, T. A 3D QSAR study of monoamino oxidase-B inhibitors using the chemical function based pharmacophore generation approach. *J Enzyme Inhib*. **2001**, 16(3), 199-215.

Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C. and Aspuru-Guzik, A. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv*. **2017**, DOI: 10.48550/ARXIV.41705.10843.

Guo, J., Janet, J. P., Bauer, M. R., Nittinger, E., Giblin, K. A., Papadopoulos, K., Voronov, A., Patronov, A., Engkvist, O. and Margreitter, C. DockStream: a docking wrapper to enhance de novo molecular design. *J Cheminform*. **2021**, 13(1), 89.

Hann, M. M. Molecular obesity, potency and other addictions in drug discovery. *MedChemComm*. **2011**, 2(5), 349-355.

Hansch, C. and Fujita, T. p-σ-π Analysis. A Method for the Correlation of Biological Activity and Chemical Structure. *J. Am. Chem. Soc.* **1964**, 86(8), 1616–1626.

Harrington, E. A., Bebbington, D., Moore, J., Rasmussen, R. K., Ajose-Adeogun, A. O., Nakayama, T., Graham, J. A., Demur, C., Hercend, T., Diu-Hercend, A. *et al.* VX-680, a potent

and selective small-molecule inhibitor of the Aurora kinases, suppresses tumor growth in vivo. *Nat Med*. **2004**, 10(3), 262-267.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J. *et al.* Array programming with NumPy. *Nature*. **2020**, 585(7825), 357-362.

Harrison, R. K. Phase II and phase III failures: 2013-2015. *Nat Rev Drug Discov*. **2016**, 15(12), 817-818.

Hartenfeller, M. and Schneider, G. De novo drug design. *Methods Mol Biol*. **2011**, 672, 299-323.

Hartenfeller, M., Zettl, H., Walter, M., Rupp, M., Reisen, F., Proschak, E., Weggen, S., Stark, H. and Schneider, G. DOGS: reaction-driven de novo design of bioactive compounds. *PLoS Comput Biol*. **2012**, 8(2), e1002380.

Hochuli, J., Helbling, A., Skaist, T., Ragoza, M. and Koes, D. R. Visualizing convolutional neural network protein-ligand scoring. *J Mol Graph Model*. **2018**, 84, 96-108.

Hopkins, A. L., Groom, C. R. and Alex, A. Ligand efficiency: a useful metric for lead selection. *Drug Discov Today*. **2004**, 9(10), 430-431.

Hopkins, A. L., Keserü, G. M., Leeson, P. D., Rees, D. C. and Reynolds, C. H. The role of ligand efficiency metrics in drug discovery. *Nat Rev Drug Discov*. **2014**, 13(2), 105-121.

Huang, S. Y., Grinter, S. Z. and Zou, X. Scoring functions and their evaluation methods for protein-ligand docking: recent advances and future directions. *Phys Chem Chem Phys*. **2010**, 12(40), 12899-12908.

Huang, S. Y. and Zou, X. Inclusion of solvation and entropy in the knowledge-based scoring function for protein-ligand interactions. *J Chem Inf Model*. **2010**, 50(2), 262-273.

Hughes, B. 2007 FDA drug approvals: a year of flux. *Nat Rev Drug Discov*. **2008**, 7(2), 107-109.

Hughes, J. P., Rees, S., Kalindjian, S. B. and Philpott, K. L. Principles of early drug discovery. *Br J Pharmacol*. **2011**, 162(6), 1239-1249.

Imrie, F., Bradley, A. R., van der Schaar, M. and Deane, C. M. Protein Family-Specific Models Using Deep Neural Networks and Transfer Learning Improve Virtual Screening and Highlight the Need for More Data. *J Chem Inf Model*. **2018**, 58(11), 2319-2330.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S. and Coleman, R. G. ZINC: a free tool to discover chemistry for biology. *J Chem Inf Model*. **2012**, 52(7), 1757-1768.

Jain, K. K. New routes for drug discovery. *Drug Discov Today*. **2002**, 7(17), 900-902.

Jeon, W. and Kim, D. Autonomous molecule generation using reinforcement learning and docking to develop potential novel inhibitors. *Sci Rep*. **2020**, 10(1), 22104.

Jin, W., Barzilay, R. and Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. *arXiv*. **2018**, DOI: 10.48550/ARXIV.41802.04364.

Jin, Z., Du, X., Xu, Y., Deng, Y., Liu, M., Zhao, Y., Zhang, B., Li, X., Zhang, L., Peng, C. *et al.* Structure of M(pro) from SARS-CoV-2 and discovery of its inhibitors. *Nature*. **2020**, 582(7811), 289-293.

Jones, G., Willett, P., Glen, R. C., Leach, A. R. and Taylor, R. Development and validation of a genetic algorithm for flexible docking. *J Mol Biol*. **1997**, 267(3), 727-748.

Kadukova, M. and Grudinin, S. Convex-PL: a novel knowledge-based potential for protein-ligand interactions deduced from structural databases using convex optimization. *Journal of Computer-Aided Molecular Design*. **2017**, 31(10), 943-958.

Kadurin, A., Aliper, A., Kazennov, A., Mamoshina, P., Vanhaelen, Q., Khrabrov, K. and Zhavoronkov, A. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*. **2017**, 8(7), 10883-10890.

Kadurin, A., Nikolenko, S., Khrabrov, K., Aliper, A. and Zhavoronkov, A. druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico. *Mol Pharm*. **2017**, 14(9), 3098-3104.

Kaelbling, L. P., Littman, M. L. and Moore, A. W. Reinforcement Learning: A Survey. *J. Artif. Intell. Res.* **1996**, 4, 237-285.

Kaserer, T., Beck, K. R., Akram, M., Odermatt, A. and Schuster, D. Pharmacophore Models and Pharmacophore-Based Virtual Screening: Concepts and Applications Exemplified on Hydroxysteroid Dehydrogenases. *Molecules*. **2015**, 20(12), 22799-22832.

Kayser, O. and Warzecha, H. *Pharmaceutical Biotechnology, Drug Discovery and Clinical Applications* (Vol. Second). Weinheim, Germany; Wiley-Blackwell, 2012.

Keller, G. Embryonic stem cell differentiation: emergence of a new era in biology and medicine. *Genes Dev*. **2005**, 19(10), 1129-1155.

Khamis, M. A., Gomaa, W. and Ahmed, W. F. Machine learning in computational docking. *Artif Intell Med*. **2015**, 63(3), 135-152.

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., Thiessen, P. A., Yu, B. *et al.* PubChem in 2021: new data content and improved web interfaces. *Nucleic Acids Res*. **2021**, 49(D1), D1388-d1395.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv*. **2014**, DOI: 10.48550/ARXIV.41412.46980.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. *arXiv*. **2014**, DOI: 10.48550/arXiv.41312.46114.

Kingma, D. P. and Welling, M. An Introduction to Variational Autoencoders. *Found. Trends Mach. Learn.* **2019**, 12(4), 307-392.

Klon, A. E., Glick, M., Thoma, M., Acklin, P. and Davies, J. W. Finding more needles in the haystack: A simple and efficient method for improving high-throughput docking results. *J Med Chem*. **2004**, 47(11), 2743-2749.

KMR-Group. (2012). Probability of Success By Molecule Size: Large vs. Small Molecules [Press release]. https://www.kmrgroup.com/wp-content/themes/responsivepro-child/PressReleases/2012_09_27%20KMR%20PBF%20Success%20Rates%20by%20Molecule%20Size%20Press%20Release.pdf (accessed November 10, 2022).

Korshunova, M., Huang, N., Capuzzi, S., Radchenko, D. S., Savych, O., Moroz, Y. S., Wells, C. I., Willson, T. M., Tropsha, A. and Isayev, O. Generative and reinforcement learning approaches for the automated de novo design of bioactive compounds. *Commun Chem*. **2022**, 5(1), 129.

Kramer, C. and Gedeck, P. Leave-cluster-out cross-validation is appropriate for scoring functions derived from diverse protein data sets. *J Chem Inf Model*. **2010**, 50(11), 1961-1969.

Kriegeskorte, N. and Golan, T. Neural network models and deep learning. *Curr Biol*. **2019**, 29(7), R231-r236.

Kulik, H. J. Large-scale QM/MM free energy simulations of enzyme catalysis reveal the influence of charge transfer. *Phys Chem Chem Phys*. **2018**, 20(31), 20650-20660.

Kurach, K., Lucic, M., Zhai, X., Michalski, M. and Gelly, S. The GAN Landscape: Losses, Architectures, Regularization, and Normalization. *arXiv*. **2018**, arXiv:1807.04720v04721.

Kutchukian, P. S., Vasilyeva, N. Y., Xu, J., Lindvall, M. K., Dillon, M. P., Glick, M., Coley, J. D. and Brooijmans, N. Inside the mind of a medicinal chemist: the role of human bias in compound prioritization during drug discovery. *PLoS One*. **2012**, 7(11), e48476.

Landrum, G. (2019). RDKit: Open-source cheminformatics. http://www.rdkit.org (accessed December 5, 2020).

Langley, J. N. On the reaction of cells and of nerve-endings to certain poisons, chiefly as regards the reaction of striated muscle to nicotine and to curari. *J Physiol*. **1905**, 33(4-5), 374-413.

Lavecchia, A. Deep learning in drug discovery: opportunities, challenges and future prospects. *Drug Discov Today*. **2019**, 24(10), 2017-2032.

Lawrence, S. Billion dollar babies--biotech drugs as blockbusters. *Nat Biotechnol*. **2007**, 25(4), 380-382.

Leach, A. R. and Gillet, V. J. *An Introduction to Cheminformatics: Revised Edition*. Dordrecht, The Netherlands; Springer 2007.

LeCun, Y., Bengio, Y. and Hinton, G. Deep learning. *Nature*. **2015**, 521(7553), 436-444.

Leung, S., Bodkin, M., von Delft, F., Brennan, P. and Morris, G. SuCOS is Better than RMSD for Evaluating Fragment Elaboration and Docking Poses. *ChemRxiv*. **2019**, DOI: 10.26434/chemrxiv.8100203.v8100201.

Leung, S. and Moesser, M. A. (2021). SuCOS. https://github.com/MarcMoesser/SuCOS (accessed March 5, 2022).

Li, S., Deng, Y. Q., Zhu, Z. L., Hua, H. L. and Tao, Z. Z. A Comprehensive Review on Radiomics and Deep Learning for Nasopharyngeal Carcinoma Imaging. *Diagnostics (Basel)*. **2021**, 11(9).

Li, Y., Liu, Z., Li, J., Han, L., Liu, J., Zhao, Z. and Wang, R. Comparative assessment of scoring functions on an updated benchmark: 1. Compilation of the test set. *J Chem Inf Model*. **2014**, 54(6), 1700-1716.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R. and Battaglia, P. Learning Deep Generative Models of Graphs. *CoRR*. **2018**, DOI: 10.48550/ARXIV.41803.03324.

Li, Y., Zhang, L. and Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *J Cheminform*. **2018**, 10(1), 33.

Lipinski, C. A., Lombardo, F., Dominy, B. W. and Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv Drug Deliv Rev*. **2001**, 46(1-3), 3-26.

Liu, H., Duan, Z., Luo, Q. and Shi, Y. Structure-based ligand design by dynamically assembling molecular building blocks at binding site. *Proteins*. **1999**, 36(4), 462-470.

Lobanovska, M. and Pilla, G. Penicillin's Discovery and Antibiotic Resistance: Lessons for the Future? *Yale J Biol Med*. **2017**, 90(1), 135-145.

Luo, Z., Wang, R. and Lai, L. RASSE: a new method for structure-based drug design. *J Chem Inf Comput Sci*. **1996**, 36(6), 1187-1194.

Lyu, J., Wang, S., Balius, T. E., Singh, I., Levit, A., Moroz, Y. S., O'Meara, M. J., Che, T., Algaa, E., Tolmachova, K. *et al.* Ultra-large library docking for discovering new chemotypes. *Nature*. **2019**, 566(7743), 224-229.

Maggiora, G., Vogt, M., Stumpfe, D. and Bajorath, J. Molecular similarity in medicinal chemistry. *J Med Chem*. **2014**, 57(8), 3186-3204.

Malhotra, S. and Karanicolas, J. When Does Chemical Elaboration Induce a Ligand To Change Its Binding Mode? *J Med Chem*. **2017**, 60(1), 128-145.

Manisha, D., Sanjeev, K. and Singh, K. K. ECG Beat Classifiers: A Journey from ANN To DNN. *Procedia Computer Science*. **2020**, 167, 747-759.

McInnes, L., Healy, J. and Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv*. **2018**, DOI: 10.48550/ARXIV.41802.03426.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. Paper presented at the Proceedings of the 9th Python in Science Conference.

Meanwell, N. A. The Influence of Bioisosteres in Drug Design: Tactical Applications to Address Developability Problems. *Tactics in Contemporary Drug Design*. **2015**, 9, 283–381.

Meli, R., Anighoro, A., Bodkin, M. J., Morris, G. M. and Biggin, P. C. Learning protein-ligand binding affinity with atomic environment vectors. *J Cheminform*. **2021**, 13(1), 59.

Mendez, D., Gaulton, A., Bento, A. P., Chambers, J., De Veij, M., Félix, E., Magariños, M. P., Mosquera, J. F., Mutowo, P., Nowotka, M. *et al.* ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res*. **2019**, 47(D1), D930-d940.

Méndez-Lucio, O., Baillif, B., Clevert, D. A., Rouquié, D. and Wichard, J. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nat Commun*. **2020**, 11(1), 10.

Meng, E. C., Shoichet, B. K. and Kuntz, I. D. Automated Docking with Grid-Based Energy Evaluation *J. Comput. Chem.* **1992**, 13(4), 505-524.

Metz, L., Ibarz, J., Jaitly, N. and Davidson, J. Discrete Sequential Prediction of Continuous Actions for Deep RL. *arXiv*. **2017**, DOI: 10.48550/ARXIV.41705.05035.

Michel, J., Tirado-Rives, J. and Jorgensen, W. L. Prediction of the water content in protein binding sites. *J Phys Chem B*. **2009**, 113(40), 13337-13346.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. *et al.* Human-level control through deep reinforcement learning. *Nature*. **2015**, 518(7540), 529-533.

Moerland, T. M., Broekens, J., Plaat, A. and Jonker, C. M. Model-based Reinforcement Learning: A Survey. *arXiv*. **2020**, DOI: 10.48550/arXiv.42006.16712.

Moesser, M. A. (2022). SARS-CoV-2-Modelling. https://github.com/MarcMoesser/SARS-CoV-2-Modelling (accessed June 24, 2022).

Moesser, M. A., Klein, D., Boyles, F., Deane, C. M., Baxter, A. and Morris, G. M. Protein-Ligand Interaction Graphs: Learning from Ligand-Shaped 3D Interaction Graphs to Improve Binding Affinity Prediction. *bioRxiv*. **2022**, DOI: 10.1101/2022.1103.1104.483012.

Moffat, J. G., Vincent, F., Lee, J. A., Eder, J. and Prunotto, M. Opportunities and challenges in phenotypic drug discovery: an industry perspective. *Nat. Rev. Drug Discov.* **2017**, 16, 531-543.

Moore, F. J. *A History of Chemistry*. New York; McGraw-Hill, 1918.

Morales, M. *Grokking Deep Reinforcement Learning*; Manning, 2020.

Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, 5(2), 107–113.

Morris, G. M., Goodshell, D. S., Pique, M. E., Lindstrom, W. L., Huey, R., Forli, S., Hart, W. E., Halliday, S., Belew, R. and Olson, A. J. *User Guide: AutoDock Version 4.2* Scripps Research Institute 2014. https://autodocksuite.scripps.edu/wp-content/uploads/sites/31/2019/03/AutoDock4.2.6_UserGuide.pdf (accessed June 22, 2023).

Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S. and Olson, A. J. AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *J Comput Chem*. **2009**, 30(16), 2785-2791.

Morse, J. S., Lalonde, T., Xu, S. and Liu, W. R. Learning from the Past: Possible Urgent Prevention and Treatment Options for Severe Acute Respiratory Infections Caused by 2019-nCoV. *Chembiochem*. **2020**, 21(5), 730-738.

Mouchlis, V. D., Afantitis, A., Serra, A., Fratello, M., Papadiamantis, A. G., Aidinis, V., Lynch, I., Greco, D. and Melagraki, G. Advances in de Novo Drug Design: From Conventional to Machine Learning Methods. *Int J Mol Sci*. **2021**, 22(4).

Murphy, K. P. *Machine Learning: A Probabilistic Perspective*; The MIT Press, 2012.

Murray, C. W., Auton, T. R. and Eldridge, M. D. Empirical scoring functions. II. The testing of an empirical scoring function for the prediction of ligand-receptor binding affinities and the use of Bayesian regression to improve the quality of the model. *Journal of Computer-Aided Molecular Design*. **1998**, 12(5), 503-519.

Newman, D. J., Cragg, G. M. and Snader, K. M. The influence of natural products upon drug discovery. *Nat Prod Rep*. **2000**, 17(3), 215-234.

Nicolaou, C. A., Kannas, C. and Loizidou, E. Multi-objective optimization methods in de novo drug design. *Mini Rev Med Chem*. **2012**, 12(10), 979-987.

Nicolotti, O., Giangreco, I., Introcaso, A., Leonetti, F., Stefanachi, A. and Carotti, A. Strategies of multi-objective optimization in drug discovery and development. *Expert Opin Drug Discov*. **2011**, 6(9), 871-884.

Nishibata, Y. and Itai, A. Automatic creation of drug candidate structures based on receptor structure. Starting point for artificial lead generation. *Tetrahedron* **1991**, 47(43), 8985-8990.

Nwaka, S. and Ridley, R. G. Virtual drug discovery and development for neglected diseases through public-private partnerships. *Nat Rev Drug Discov*. **2003**, 2(11), 919-928.

O'Boyle, N. M. Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI. *J Cheminform*. **2012**, 4(1), 22.

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T. and Hutchison, G. R. Open Babel: An open chemical toolbox. *J Cheminform*. **2011**, 3, 33.

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T. and Hutchison, G. R. (2020). Open Babel: The Open Source Chemistry Toolbox. http://openbabel.org/wiki/Main_page (accessed December 5, 2020).

O'Boyle, N. M. and Sayle, R. A. Comparing structural fingerprints using a literature-based similarity benchmark. *J Cheminform*. **2016**, 8, 36.

Olivecrona, M., Blaschke, T., Engkvist, O. and Chen, H. Molecular de-novo design through deep reinforcement learning. *J Cheminform*. **2017**, 9(1), 48.

Orozco-Gonzalez, Y., Manathunga, M., Marín, M. D. C., Agathangelou, D., Jung, K. H., Melaccio, F., Ferré, N., Haacke, S., Coutinho, K., Canuto, S. *et al.* An Average Solvent Electrostatic Configuration Protocol for QM/MM Free Energy Optimization: Implementation and Application to Rhodopsin Systems. *J Chem Theory Comput*. **2017**, 13(12), 6391-6404.

Osband, I., Blundell, C., Pritzel, A. and Van Roy, B. Deep Exploration via Bootstrapped DQN. *arXiv*. **2016**, DOI: 10.48550/arXiv.41602.04621.

Osband, I., Van Roy, B., Russo, D. J. and Wen, Z. Deep Exploration via Randomized Value Functions. *JMLR*. **2019**, 20, 1-62.

Palazzesi, F. and Pozzan, A. (2022). Deep Learning Applied to Ligand-Based De Novo Drug DesignDe novo drug design. In A. Heifetz (Ed.), Artificial Intelligence in Drug Design (pp. 273-299). New York, NY: Springer US.

Pammolli, F., Magazzini, L. and Riccaboni, M. The productivity crisis in pharmaceutical R&D. *Nat Rev Drug Discov*. **2011**, 10(6), 428-438.

Patani, G. A. and LaVoie, E. J. Bioisosterism: A Rational Approach in Drug Design. *Chem Rev*. **1996**, 96(8), 3147-3176.

Payne, A. W. and Glen, R. C. Molecular recognition using a binary genetic search algorithm. *J Mol Graph*. **1993**, 11(2), 74-91, 121-123.

Pearlman, D. A. and Murcko, M. A. CONCERTS: dynamic connection of fragments as an approach to de novo ligand design. *J Med Chem*. **1996**, 39(8), 1651-1663.

Pina, A. S., Hussain, A. and Roque, A. C. An historical overview of drug discovery. *Methods Mol Biol*. **2009**, 572, 3-12.

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M. *et al.* Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *Front Pharmacol*. **2020**, 11, 565644.

Polykovskiy, D., Zhebrak, A., Vetrov, D., Ivanenkov, Y., Aladinskiy, V., Mamoshina, P., Bozdaganyan, M., Aliper, A., Zhavoronkov, A. and Kadurin, A. Entangled Conditional Adversarial Autoencoder for de Novo Drug Discovery. *Mol Pharm*. **2018**, 15(10), 4398-4405.

Popova, M., Isayev, O. and Tropsha, A. Deep reinforcement learning for de novo drug design. *Sci Adv*. **2018**, 4(7), eaap7885.

Ragoza, M., Hochuli, J., Idrobo, E., Sunseri, J. and Koes, D. R. Protein-Ligand Scoring with Convolutional Neural Networks. *J Chem Inf Model*. **2017**, 57(4), 942-957.

Raha, K., Peters, M. B., Wang, B., Yu, N., Wollacott, A. M., Westerhoff, L. M. and Merz, K. M., Jr. The role of quantum mechanics in structure-based drug design. *Drug Discov Today*. **2007**, 12(17-18), 725-731.

Rasmussen, C. E. (2004). Gaussian Processes in Machine Learning. In O. Bousquet, U. von Luxburg and G. Rätsch (Eds.), Advanced Lectures on Machine Learning: ML Summer Schools

2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures (pp. 63-71). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ravindranath, P. A., Forli, S., Goodsell, D. S., Olson, A. J. and Sanner, M. F. AutoDockFR: Advances in Protein-Ligand Docking with Explicitly Specified Binding Site Flexibility. *PLoS Comput Biol*. **2015**, 11(12), e1004586.

Richards, A. (2015). University of Oxford Advanced Research Computing. Zenodo. https://doi.org/10.5281/zenodo.22558 (accessed September 15, 2020).

Rocke, A. J. It began with a daydream: the 150th anniversary of the Kekulé benzene structure. *Angew Chem Int Ed Engl*. **2015**, 54(1), 46-50.

Rodrigues, T., Roudnicky, F., Koch, C. P., Kudoh, T., Reker, D., Detmar, M. and Schneider, G. De novo design and optimization of Aurora A kinase inhibitors. *Chem. Sci.* **2013**, 4(3), 1229-1233.

Rogers, D. and Hahn, M. Extended-connectivity fingerprints. *J Chem Inf Model*. **2010**, 50(5), 742-754.

Roughley, S. D. and Jordan, A. M. The medicinal chemist's toolbox: an analysis of reactions used in the pursuit of drug candidates. *J Med Chem*. **2011**, 54(10), 3451-3479.

Ruiz-Montiel, M., Mandow, L. and Pérez-de-la-Cruz, J. A temporal difference method for multi-objective reinforcement learning. *Neurocomputing*. **2017**, 263, 15-25.

Russell, S. J. and Norvig, P. *Artificial Intelligence: A Modern Approach* (Third ed.); Prentice Hall, 2009.

Samanta, B., De, A., Jana, G., Gomez, V., Chattaraj, P. K., Ganguly, N. and Gomez-Rodriguez, M. NEVAE: A Deep Generative Model for Molecular Graphs. *JMLR*. **2020**, 21, 1-33.

Sanchez-Garcia, R., Havasi, D., Takács, G., Robinson, M. C., Lee, A., von Delft, F. and Deane, C. M. CoPriNet: graph neural networks provide accurate and rapid compound price prediction for molecule prioritisation. *Digital Discovery*. **2023**.

Sanchez-Lengeling, B., Outeiral, C., Guimaraes, G. L. and Aspuru-Guzik, A. Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). *ChemRxiv*. **2017**, DOI: 10.26434/chemrxiv.5309668.v5309663.

Sanchez-Lengeling, B., Wei, J., Lee, B., Reif, E., Wang, P., Qian, W., McCloskey, K., Colwell, L. and Wiltschko, A. (2020). Evaluating Attribution for Graph Neural Networks. Paper presented at the Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada.

Sartipy, P., Björquist, P., Strehl, R. and Hyllner, J. The application of human embryonic stem cell technologies to drug discovery. *Drug Discov Today*. **2007**, 12(17-18), 688-699.

Scantlebury, J., Vost, L., Carbery, A., Hadfield, T. E., Turnbull, O. M., Brown, N., Chenthamarakshan, V., Das, P., Grosjean, H., Delft, F. v. *et al.* PointVS: A Machine Learning Scoring Function that Identifies Important Binding Interactions. *bioRxiv*. **2022**, DOI: 10.1101/2022.1110.1128.511712.

Schaul, T., Quan, J., Antonoglou, I. and Silver, D. Prioritized Experience Replay. *arXiv*. **2015**, DOI: 10.48550/ARXIV.41511.05952.

Schindler, T. Roche, Basel, Switzerland. Personal Communication, 2019.

Schneider, G. and Fechner, U. Computer-based de novo design of drug-like molecules. *Nat Rev Drug Discov*. **2005**, 4(8), 649-663.

Schneider, G., Lee, M. L., Stahl, M. and Schneider, P. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. *J Comput Aided Mol Des*. **2000**, 14(5), 487-494.

Schneider, P. and Schneider, G. De Novo Design at the Edge of Chaos. *J Med Chem*. **2016**, 59(9), 4077-4086.

Schrödinger, L. (2022). The PyMOL Molecular Graphics System. https://pymol.org/2/ (accessed December 5, 2020).

Schultes, S., de Graaf, C., Haaksma, E. E. J., de Esch, I. J. P., Leurs, R. and Kramer, O. Ligand efficiency as a guide in fragment hit selection and optimization. *Drug Discov Today Technol*. **2010**, 7(3), e147-202.

Scott, A. D., Phillips, C., Alex, A., Flocco, M., Bent, A., Randall, A., O'Brien, R., Damian, L. and Jones, L. H. Thermodynamic optimisation in drug discovery: a case study using carbonic anhydrase inhibitors. *ChemMedChem*. **2009**, 4(12), 1985-1989.

Shayakhmetov, R., Kuznetsov, M., Zhebrak, A., Kadurin, A., Nikolenko, S., Aliper, A. and Polykovskiy, D. Molecular Generation for Desired Transcriptome Changes With Adversarial Autoencoders. *Front Pharmacol*. **2020**, 11, 269.

Shen, C., Ding, J., Wang, Z., Cao, D., Ding, X. and Hou, T. From machine learning to deep learning: Advances in scoring functions for protein–ligand docking. *WIREs Comput Mol Sci.* **2019**, e1429, 1-23.

Stoll, F., Göller, A. H. and Hillisch, A. Utility of protein structures in overcoming ADMET-related issues of drug-like compounds. *Drug Discov Today*. **2011**, 16(11-12), 530-538.

Su, M., Yang, Q., Du, Y., Feng, G., Liu, Z., Li, Y. and Wang, R. Comparative Assessment of Scoring Functions: The CASF-2016 Update. *J Chem Inf Model*. **2019**, 59(2), 895-913.

Sun, D., Gao, W., Hu, H. and Zhou, S. Why 90% of clinical drug development fails and how to improve it? *Acta Pharm Sin B*. **2022**, 12(7), 3049-3062.

Sun, H., Li, Y., Li, D. and Hou, T. Insight into crizotinib resistance mechanisms caused by three mutations in ALK tyrosine kinase using free energy calculation approaches. *J Chem Inf Model*. **2013**, 53(9), 2376-2389.

Sun, H. Y., Hou, T. J. and Zhang, H. Y. Finding chemical drugs for genetic diseases. *Drug Discov Today*. **2014**, 19(12), 1836-1840.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning, Second Edition: An Introduction*. Cambridge, MA; The MIT Process, 2018.

Świderek, K. and Moliner, V. Revealing the molecular mechanisms of proteolysis of SARS-CoV-2 M(pro) by QM/MM computational methods. *Chem Sci*. **2020**, 11(39), 10626-10630.

Thomas, G. *Medicinal Chemistry: An Introduction* (2nd ed.). Chichester, England; John Wiley & Sons 2007.

Thomas, M., Smith, R. T., O'Boyle, N. M., de Graaf, C. and Bender, A. Comparison of structure- and ligand-based scoring functions for deep generative models: a GPCR case study. *J Cheminform*. **2021**, 13(1), 39.

Thomson, J. A., Itskovitz-Eldor, J., Shapiro, S. S., Waknitz, M. A., Swiergiel, J. J., Marshall, V. S. and Jones, J. M. Embryonic stem cell lines derived from human blastocysts. *Science*. **1998**, 282(5391), 1145-1147.

Tokic, M. and Palm, G. (2011). Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax, Berlin, Heidelberg.

Travis, A. S. Chemical Modeling: From Paul Ehrlich's Dyes to β-Blockers-A Brief History. *J Comput Biol*. **2019**, 26(7), 726-734.

Trott, O. and Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J Comput Chem*. **2010**, 31(2), 455-461.

Tucker, J. Y., S.; Yakatan, S. Biogenerics 2007: How far have we come? *J. Commer. Biotechnol.* **2008**, 14, 56–64.

van der Maaten, L. and Hinton, G. Visualizing Data using t-SNE. *JMLR*. **2008**, 9(86), 2579-2605.

van Engelen, J. E. and Hoos, H. H. A survey on semi-supervised learning. *Machine Learning*. **2020**, 109(2), 373-440.

van Hasselt, H., Guez, A. and Silver, D. (2016). Deep reinforcement learning with double Q-Learning. Paper presented at the Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona.

Walsh, G. *Biopharmaceuticals: Biochemistry and Biotechnology* (Second ed.). Chichester, England; John Wiley & Sons, Ltd., 2003.

Walters, P. (2018). rd_filters. https://github.com/PatWalters/rd_filters (accessed November 10, 2021).

Walters, P. (2021). silly_walks. https://github.com/PatWalters/silly_walks (accessed November 23, 2021).

Wang, M., Wang, Z., Sun, H., Wang, J., Shen, C., Weng, G., Chai, X., Li, H., Cao, D. and Hou, T. Deep learning approaches for de novo drug design: An overview. *Curr Opin Struct Biol*. **2022**, 72, 135-144.

Waskom, M. L. seaborn: statistical data visualization. *J. Open Source Softw.* **2021**, 6(60), 3021.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Mach Learn*. **1992**, 8(3), 279-292.

Watson, J. D. and Crick, F. H. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*. **1953**, 171(4356), 737-738.

Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, 28(1), 31-36.

Wermuth, C. G., Ganellin, C. R., Lindberg, P. and Mitscher, L. A. Glossary of terms used in medicinal chemistry (IUPAC Recommendations 1998). *Pure Appl. Chem.* **1998**, 70(5), 1129–1143.

Wildman, S. A. and Crippen, G. M. Prediction of Physicochemical Parameters by Atomic Contributions. *J. Chem. Inf. Comput. Sci.* **1999**, 39(5), 868–873.

Wise, A., Gearing, K. and Rees, S. Target validation of G-protein coupled receptors. *Drug Discov Today*. **2002**, 7(4), 235-246.

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. *et al.* DrugBank 5.0: a major update to the DrugBank database for 2018. *Nucleic Acids Res*. **2018**, 46(D1), D1074-d1082.

Wong, C. H., Siah, K. W. and Lo, A. W. Estimation of clinical trial success rates and related parameters. *Biostatistics*. **2019**, 20(2), 273-286.

Wood, D. J., de Vlieg, J., Wagener, M. and Ritschel, T. Pharmacophore fingerprint-based approach to binding site subpocket similarity and its application to bioisostere replacement. *J Chem Inf Model*. **2012**, 52(8), 2031-2043.

Woodward, D. J., Bradley, A. R. and van Hoorn, W. P. Coverage Score: A Model Agnostic Method to Efficiently Explore Chemical Space. *J Chem Inf Model*. **2022**, 62(18), 4391-4402.

World-Health-Organization (2022). WHO Coronavirus (COVID-19) Dashboard. https://covid19.who.int/#:~:text=Globally%2C%20as%20of%204%3A38pm,6%2C630%2C082%20deaths%2C%20reported%20to%20WHO. (accessed December 13, 2022).

Xu, Y., Lin, K., Wang, S., Wang, L., Cai, C., Song, C., Lai, L. and Pei, J. Deep learning for molecular generation. *Future Med Chem*. **2019**, 11(6), 567-597.

Yang, B., Li, K., Zhong, X. and Zou, J. Implementation of deep learning in drug design. *MedComm – Future Medicine*. **2022**, 1(2), e18.

Yang, X., Wang, Y., Byrne, R., Schneider, G. and Yang, S. Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery. *Chem Rev*. **2019**, 119(18), 10520-10594.

Yang, X., Zhang, J., Yoshizoe, K., Terayama, K. and Tsuda, K. ChemTS: an efficient python library for de novo molecular generation. *Sci Technol Adv Mater*. **2017**, 18(1), 972-976.

Yang, Y., Lightstone, F. C. and Wong, S. E. Approaches to efficiently estimate solvation and explicit water energetics in ligand binding: the use of WaterMap. *Expert Opin Drug Discov*. **2013**, 8(3), 277-287.

Yasonik, J. Multiobjective de novo drug design with recurrent neural networks and nondominated sorting. *J Cheminform*. **2020**, 12(1), 14.

You, J., Liu, B., Ying, R., Pande, V. and Leskovec, J. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. *arXiv*. **2018**, DOI: 10.48550/ARXIV.41806.02473.

Zanders, E. D. (2020). Drug Discovery Pipeline Overview. In The Science and Business of Drug Discovery (pp. 95-98): Springer, Cham.

Zhang, H., Wang, Y., Sun, J. and Xu, Z. Amortized Variational Deep Q Network. *arXiv*. **2020**, DOI: 10.48550/arXiv.42011.01706.

Zhang, J., Norinder, U. and Svensson, F. Deep Learning-Based Conformal Prediction of Toxicity. *J Chem Inf Model*. **2021**, 61(6), 2648-2657.

Zhou, Z., Kearnes, S., Li, L., Zare, R. N. and Riley, P. Optimization of Molecules via Deep Reinforcement Learning. *Sci Rep*. **2019**, 9(1), 10752.

Zhu, J., Yu, H., Fan, H., Liu, H. and Shi, Y. Design of new selective inhibitors of cyclooxygenase-2 by dynamic assembly of molecular building blocks. *J Comput Aided Mol Des*. **2001**, 15(5), 447-463.

Zipser, D. and Andersen, R. A. A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*. **1988**, 331(6158), 679-684.

Appendix A: Reaction SMARTS

Table A1: Reaction SMARTS of the disconnection reactions corresponding to top 10 reactions (Roughley and Jordan, 2011). Note, ring opening disconnections are avoided by ensuring that two synthons are generated.

	Reaction Type	Disconnection Reaction SMARTS
1	Amides	[#7:1][C;x0:2]=[O:3]>>[#7:1].Cl[C:2]=[O:3]
2	N-Arylation	[CX4;H0:2]-[#7X3;R0:1]>>[#7:1].[#6:2]Br [CX4;H1:2]-[#7X3;R0:1]>>[#7:1].[#6:2]Br [CX4;H2R0:2]-[#7X3:1]>>[#7:1].[#6:2]Br [CX4;H1R0:2]-[#7X3:1]>>[#7:1].[#6:2]Br [CX4;H2:2]-[#7X3;R0:1]>>[#7:1].[#6:2]Br [CX4;H3:2]-[#7X3;R0:1]>>[#7:1].[#6:2]Br [CX4;H0R0:2]-[#7X3:1]>>[#7:1].[#6:2]Br
3	S_NAr	[#6aR1:2]!@[NX3;H0,H1:1][#6:3]>>[#6:3][#7X3;H1,H2:1].[#6aR1:2]!@Br
4	Reductive Amination	[CX4;H1:2]-[#7X3;R0:1]>>[#7X3:1].[#6X4;H0:2]=O [CX4;H2R0:2]-[#7X3:1]>>[#7X3:1].[#6X4;H0:2]=O [CX4;H1R0:2]-[#7X3:1]>>[#7X3:1].[#6X4;H0:2]=O [CX4;H2:2]-[#7X3;R0:1]>>[#7X3:1].[#6:2]=O [CX4;H3:2]-[#7X3;R0:1]>>[#7X3:1].[#6:2]=O
5	Ester Coupling	[#8:1][C;x0:2]=[O:3]>>[#8:1].Cl[C:2]=[O:3]
5	Ether Coupling	[CX4;H0:2]-[#8X2;R0:1]>>[#8:1].[#6:2]Br [CX4;H1:2]-[#8X2;R0:1]>>[#8:1].[#6:2]Br [CX4;H2R0:2]-[#8X2:1]>>[#8:1].[#6:2]Br [CX4;H1R0:2]-[#8X2:1]>>[#8:1].[#6:2]Br [CX4;H2:2]-[#8X2;R0:1]>>[#8:1].[#6:2]Br [CX4;H3:2]-[#8X2;R0:1]>>[#8:1].[#6:2]Br [CX4;H0R0:2]-[#8X2:1]>>[#8:1].[#6:2]Br [n:3][c:2]-[#8X2:1]>>[#8:1].[n:3][#6:2]Br
6	Sonogashira	[#6aR1:1]!@[C:2]#[C:3]>>[#6aR1:1]Br.[C:2]#[C:3]
7	Suzuki Coupling	[#6aR1:1]!@[#6aR1:2]>>[#6aR1:2]Br.[#6aR1:1][#5](-[#8])-[#8]
8	Urea	[#7:1][C;x0]([#7:2])=O>>[#7:1].[#7:2]
9	Sulfonamide	[#7:1][S:2](=[O:3])=[O:4]>>[#7:1].Cl[S:2](=[O:3])=[O:4]

Table A2: Reaction SMARTS of the forward reactions corresponding to top 10 reactions (Roughley and Jordan, 2011).

	Reaction	Varied Synthon	Reaction SMARTS
1	Amides	Acids	[Cl,Br,I][C:2]=[O:3].[#7:1]>>[#7:1][C:2]=[O:3] [OH1][C:2]=[O:3].[#7:1]>>[#7:1][C:2]=[O:3]
		Amines	[NH2:1].Cl[C:2]=[O:3]>>[#7:1][C:2]=[O:3] [NH1:1].Cl[C:2]=[O:3]>>[#7:1][C:2]=[O:3]
2	N-Arylation	Amines	[#7X3;H2:1].[#6;A:2]Br>>[#7:1]-[#6:2] [#7X3;H1:1].[#6;A:2]Br>>[#7:1]-[#6:2]
		Halides	[C:2][Cl,Br,I,F].[#7X3;H2:1]>>[#7:1]-[#6:2] [C:2][Cl,Br,I,F].[#7X3;H1:1]>>[#7:1]-[#6:2]
3	S_NAr	Amines	[#7H2:1].[c:2][Br,Cl,I,F]>>[c:2][NH1:1] [NH1:1].[c:2][Br,Cl,I,F]>>[c:2][NH0:1]
		Aromatic Halides	[c:2]-[Cl,Br,I,F].[#7:1]>>[c:2]-[#7:1]
4	Reductive Amination	Ketones/ Aldehydes	[#6:3][CX3:2](=O)[#6:4].[#7X3;H1:1]>>[#6:3]-[#6:2](-[#6:4])-[#7:1] [CX3;H1:2](=O)[#6:3].[#7X3;H1:1]>>[#6:3]-[#6:2]-[#7:1]
		Amines	[#7X3;AH2:1].[#6:4][CX3:2](=O)[#6:3]>>[#6:4]-[#6:2](-[#6:3])-[#7:1] [#7X3;AH1:1].[#6:2][CX3:3](=O)[#6:4]>>[#6:2]-[#6:3](-[#6:4])-[#7:1] [#7X3;AH2:1].[CX3;H1:2](=O)[#6:4]>>[#6:4]-[#6:2]-[#7:1] [#7X3;AH1:1].[CX3;H1:3](=O)[#6:2]>>[#6:2]-[#6:3]-[#7:1]
5	Ester Coupling	Acids	[Cl,Br,I,F][C:2]=[O:3].[OX2;H1:1]>>[#8:1][C:2]=[O:3]
		Alcohols	[OH1][C:2]=[O:3].[OX2;H1:1]>>[#8:1][C:2]=[O:3] [OX2;H1:1].Cl[C:2]=[O:3]>>[#8:1][C:2]=[O:3]
5	Ether Coupling	Alcohols	[#8X2;H1:1].[#6;A:2]Br>>[#6:2]-[#8:1] [OX2;H1:1].[n:3][#6;a:2]Br>>[O:1]-[#6:2][n:3]
		Halides	[#6X4;A:2][#9, #17,#35,#53].[#8X2;H1:1]>>[#6:2][#8:1] [n:3][c:2][#9,#17,#35,#53].[#8X2;H1:1]>>[n:3][c:2][#8:1]
6	Sonogashira	Alkynes	[CH1:1]#[C:2].[#6aR1:3]-[#9,#17,#35,#53] >>[#6aR1:3]!@[CH0:1]#[C:2]
		Aromatic Halides	[#6aR1:1][#9,#17,#35,#53].[CH1:2]#[C:3]>>[#6aR1:1]!@[CH0:2]#[C:3]
7	Suzuki Coupling	Aromatic Halides	[#6aR1:2]-[#9,#17,#35,#53].[#6aR1:1]-[#5](-[#8])-[#8] >>[#6aR1:1]!@[#6aR1:2]
		Boronic Acids	[#6aR1:1]-[#5](-[#8])-[#8].[#6aR1:2]- [#9,#17,#35,#53]>>[#6aR1:1]!@[#6aR1:2]
8	Urea	Amines	[#7X3;H2:1].[#7:2]>>[#7X3;H1:1]C([#7:2])=O [NX3;H1:1].[#7:2]>>[#7X3;H0:1]C([#7:2])=O
		Isocyanates	[#7:2]=C=O.[#7X3;H2:1]>>[#7:1]-[#6](-[#7:2])=O [#7:2]=C=O.[#7;AH1:1]>>[#7:1]-[#6](-[#7:2])=O [#7X3;H2:1][C:2]=[O:3]>>[#7X3;H1:1]=[C:2]=[O:3] [#7X3;H1:1][C:2]=[O:3]>>[#7X3;H0:1]=[C:2]=[O:3]
9	Sulfonamide	Sulfonyl Chlorides	[#17,#35,#53][S:4](=[O:3])=[O:2].[#7:1]>>[#7:1][S:4](=[O:3])=[O:2]
		Amines	[#7X3;H2:1].[Cl,Br,I,F][S:2](=[O:3])=[O:4]>>[#7X3:1][S:2](=[O:3])=[O:4] [#7X3;H1:1].[Cl,Br,I,F][S:2](=[O:3])=[O:4]>>[#7X3:1][S:2](=[O:3])=[O:4]

Table A3: SMARTS reactions used to generate compound amino acid adduct from new molecules generated using c3D-DQN methods based on covalent warhead present in X-ray crystal structures of SARS-CoV-2 main protease from Fragalysis.

Covalent Warhead	Reaction SMARTS
Acrylamide	[SX2;H1:3][#6X4;H2:1][#6X4;H1:2]([NX3;H2:5])[#6X3;H1:4](=[O:6]).[#7X3:8][CX3:9](=[O:10])[C:11]=[C:12]>>[O:6]=[#6X3;H1:4][#6X4;H1:2]([NX3;H2:5])[#6X4;H2:1][SX2;H0:3][C:12][C:11][C;x0:9]([#7X3:8])=[O:10] [SX2;H1:3][#6X4;H2:1][#6X4;H1:2]([NX3;H2:5])[#6X3;H1:4](=[O:6]).[#7X3:8][CX3:9](=[O:10])[C:11][C:12]>>[O:6]=[#6X3;H1:4][#6X4;H1:2]([NX3;H2:5])[#6X4;H2:1][SX2;H0:3][C:12][C:11][C;x0:9]([#7X3:8])=[O:10]
Chloroacetamide	[SX2;H1:3][#6X4;H2:1][#6X4;H1:2]([NX3;H2:5])[#6X3;H1:4](=[O:6]).[#7X3:8][CX3:9](=[O:10])[C:11][Cl:12]>>[O:6]=[#6X3;H1:4][#6X4;H1:2]([NX3;H2:5])[#6X4;H2:1][SX2;H0:3][C:11][C;x0:9]([#7X3:8])=[O:10] [SX2;H1:3][#6X4;H2:1][#6X4;H1:2]([NX3;H2:5])[#6X3;H1:4](=[O:6]).[#7X3:8][CX3:9](=[O:10])[C:11]>>[O:6]=[#6X3;H1:4][#6X4;H1:2]([NX3;H2:5])[#6X4;H2:1][SX2;H0:3][C:11][C;x0:9]([#7X3:8])=[O:10]
Nitrile	[SX2;H1:3][#6X4;H2:1][#6X4;H1:2]([NX3;H2:5])[#6X3;H1:4](=[O:6]).[NX1:8]#[CX2:9]>>[O:6]=[#6X3;H1:4][#6X4;H1:2]([NX3;H2:5])[#6X4;H2:1][SX2;H0:3][CX3:9]=[NX2;H1:8] [SX2;H1:3][#6X4;H2:1][#6X4;H1:2]([NX3;H2:5])[#6X3;H1:4](=[O:6]).[NX2;H1:8]=[CX3:9]>>[O:6]=[#6X3;H1:4][#6X4;H1:2]([NX3;H2:5])[#6X4;H2:1][SX2;H0:3][CX3:9]=[NX2;H1:8]
Vinylsulfoamide	[SX2;H1:3][#6X4;H2:1][#6X4;H1:2]([NX3;H2:5])[#6X3;H1:4](=[O:6]).[#16X4:8]([NX3:9])(=[OX1:10])(=[OX1:11])[#6X3:12]=[#6X3:13]>>[O:6]=[#6X3;H1:4][#6X4;H1:2]([NX3;H2:5])[#6X4;H2:1][SX2;H0:3][#6X4:13][#6X4:12][#16X4:8](=[OX1:11])(=[OX1:10])([NX3:9]) [SX2;H1:3][#6X4;H2:1][#6X4;H1:2]([NX3;H2:5])[#6X3;H1:4](=[O:6]).[#16X4:8]([NX3:9])(=[OX1:10])(=[OX1:11])[#6X4:12][#6X4:13]>>[O:6]=[#6X3;H1:4][#6X4;H1:2]([NX3;H2:5])[#6X4;H2:1][SX2;H0:3][#6X4:13][#6X4:12][#16X4:8](=[OX1:11])(=[OX1:10])([NX3:9])

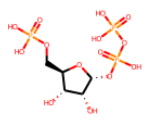
Table A4: SMARTS used for masking the covalent warhead from the starting molecule for SARS-CoV-2 main protease from Fragalysis in the c3D-DQN methods.

Covalent Warhead	SMARTS for Masking
Acrylamide	[#7X3:8][CX3:9](=[OX1:10])[CX4;H2:11][CX4;H3:12]
Chloroacetamide	[#7X3:8][CX3:9](=[OX1:10])[CX4;H2:11][Cl:12]
Nitrile	[NX1:8]#[CX2;H0:9]
Vinylsulfoamide	[#16X4:8]([NX3:9])(=[OX1:10])(=[OX1:11])[#6X3:12]=[#6X3:13]

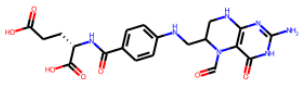
Appendix B: Reagents and Starting Molecules

Table B1: Number of reagents for each reaction from Reaxys including seed, which was used to test 100 chemically diverse drugs from DrugBank in 2D-DQN methods (Wishart *et al.*, 2018).

Reactions	Varied Synthon	All Reaxys Reagents	Selected Reaxys Reagents
Amides	Acids	2232	109
	Amines	10503	98
N-Arylation	Amines	7211	130
	Halides	1518	92
S_NAr	Amines	7302	73
	Aromatic Halides	8645	71
Reductive Amination	Ketones/ Aldehydes	2072	137
	Amines	3103	130
Ester Coupling	Acids	448	52
	Alcohols	7626	104
Ether Coupling	Alcohols	6569	74
	Halides	3674	170
Sonogashira	Alkynes	2066	50
	Aromatic Halides	3652	50
Suzuki Coupling	Aromatic Halides	2899	67
	Boronic Acids	1018	68
Urea	Amines	2972	51
	Isocyanates	504	51
Sulfonamide	Sulfonyl Chlorides	714	61
	Amines	2982	59



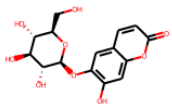
5-O-phosphono-alpha-D-ribofuranosyl diphosphate / DB01632



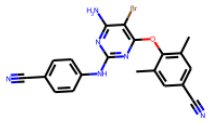
Leucovorin / DB00650



Flucloxacillin / DB00301



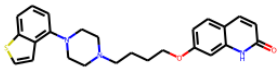
Esculin / DB13155



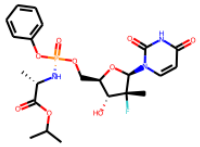
Etravirine / DB06414



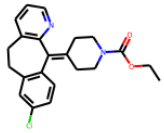
Pivampicillin / DB01604



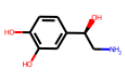
Brexpiprazole / DB09128



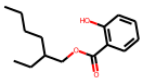
Sofosbuvir / DB08934



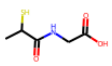
Loratadine / DB00455



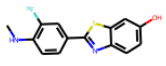
Norepinephrine / DB00368



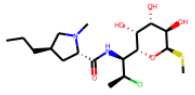
Octisalate / DB11062



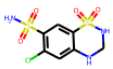
Tiopronin / DB06823



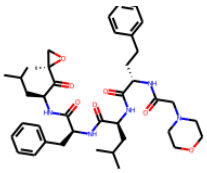
Flutemetamol (18F) / DB09151



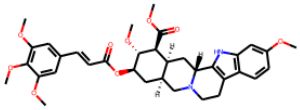
Clindamycin / DB01190



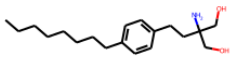
Hydrochlorothiazide / DB00999



Carfilzomib / DB08889



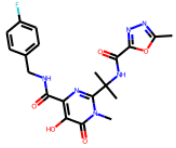
Rescinnamine / DB01180



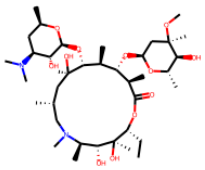
Fingolimod / DB08868



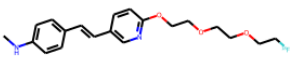
Propylhexedrine / DB06714



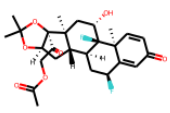
Raltegravir / DB06817



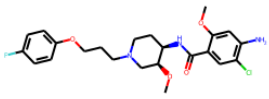
Azithromycin / DB00207



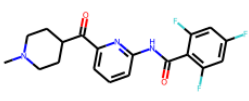
Florbetapir (18F) / DB09149



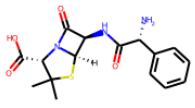
Fluocinonide / DB01047



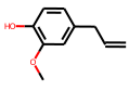
Cisapride / DB00604



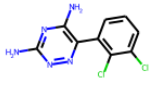
Lasmiditan / DB11732



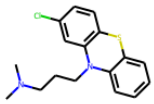
Ampicillin / DB00415



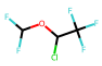
Eugenol / DB09086



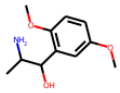
Lamotrigine / DB00555



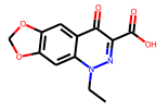
Chlorpromazine / DB00477



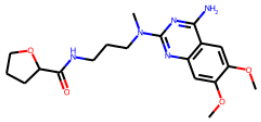
Isoflurane / DB00753



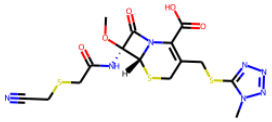
Methoxamine / DB00723



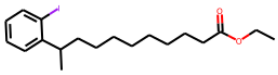
Cinoxacin / DB00827



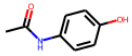
Alfuzosin / DB00346



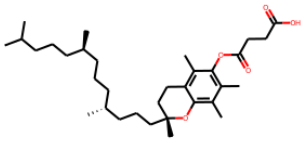
Cefmetazole / DB00274



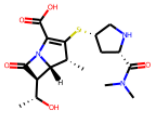
Iophendylate / DB01187



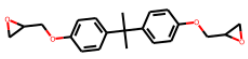
Acetaminophen / DB00316



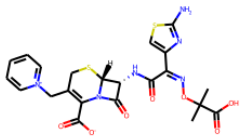
alpha-Tocopherol succinate / DB14001



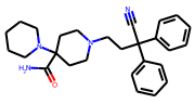
Meropenem / DB00760



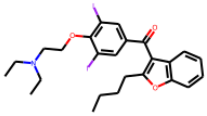
Bisphenol A diglycidyl ether / DB14083



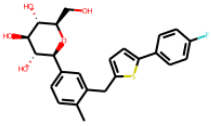
Ceftazidime / DB00438



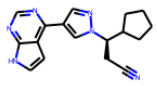
Piritramide / DB12492



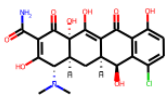
Amiodarone / DB01118



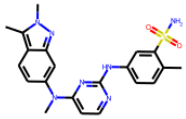
Canagliflozin / DB08907



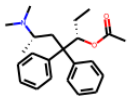
Ruxolitinib / DB08877



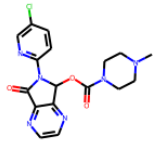
Demeclocycline / DB00618



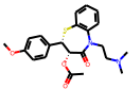
Pazopanib / DB06589



Levacetylmethadol / DB01227



Zopiclone / DB01198



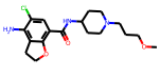
Diltiazem / DB00343



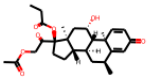
Tranylcypromine / DB00752



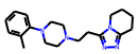
Eslicarbazepine acetate / DB09119



Prucalopride / DB06480



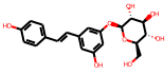
Methylprednisolone aceponate / DB14643



Dapiprazole / DB00298



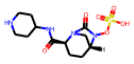
Aceclidine / DB13262



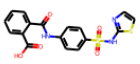
Polydatin / DB11263



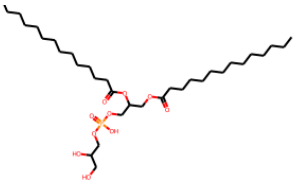
Phenylpropanolamine / DB00397



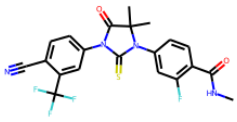
Relebactam / DB12377



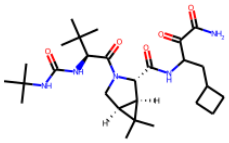
Phthalylsulfathiazole / DB13248



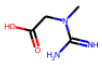
DL-dimyristoylphosphatidylglycerol / DB1128



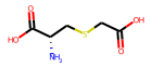
Enzalutamide / DB08899



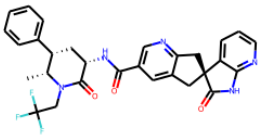
Boceprevir / DB08873



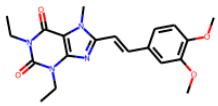
Creatine / DB00148



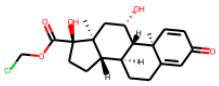
Carbocisteine / DB04339



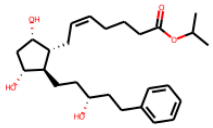
Ubrogepant / DB15328



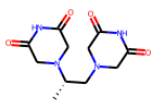
Istradefylline / DB11757



Loteprednol / DB00873



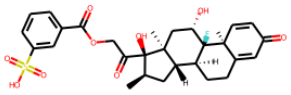
Latanoprost / DB00654



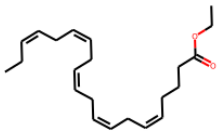
Dexrazoxane / DB00380



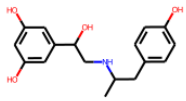
Framycetin / DB00452



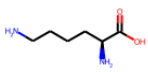
Dexamethasone metasulfobenzoate / DB14703



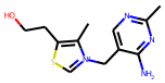
Icosapent ethyl / DB08887



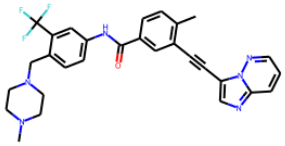
Fenoterol / DB01288



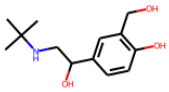
L-Lysine / DB00123



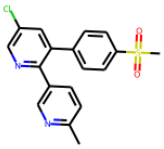
Thiamine / DB00152



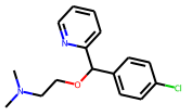
Ponatinib / DB08901



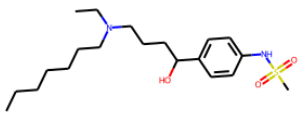
Salbutamol / DB01001



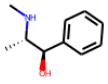
Etoricoxib / DB01628



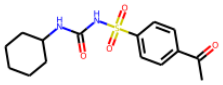
Carbinoxamine / DB00748



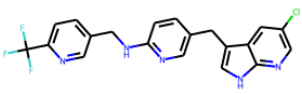
Ibutilide / DB00308



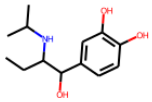
Ephedrine / DB01364



Acetohexamide / DB00414



Pexidartinib / DB12978



Isoetharine / DB00221

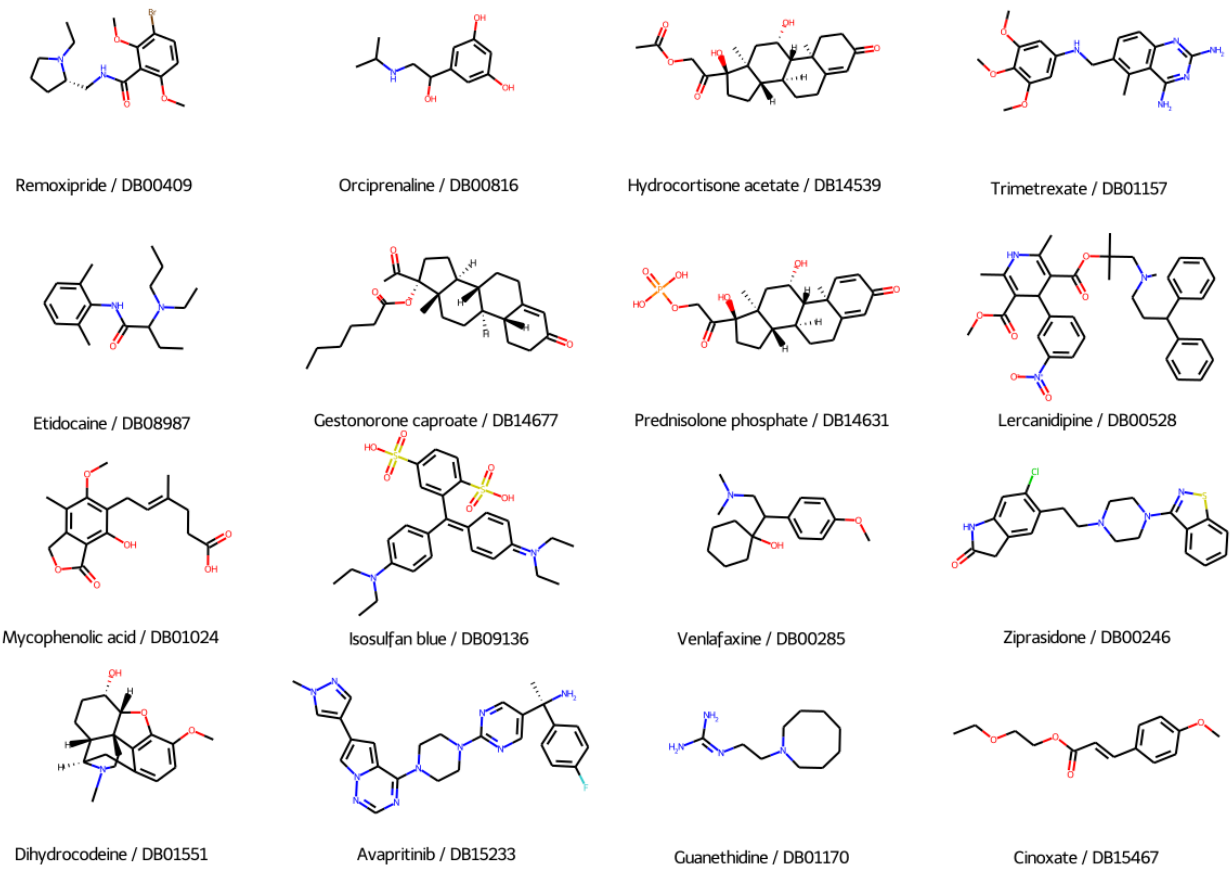


Figure B2: 100 Starting DrugBank (Wishart *et al.*, 2018) molecules used to test MolDQN, RxnDQN, and RxnMolDQN with their drug name and DrugBank ID.

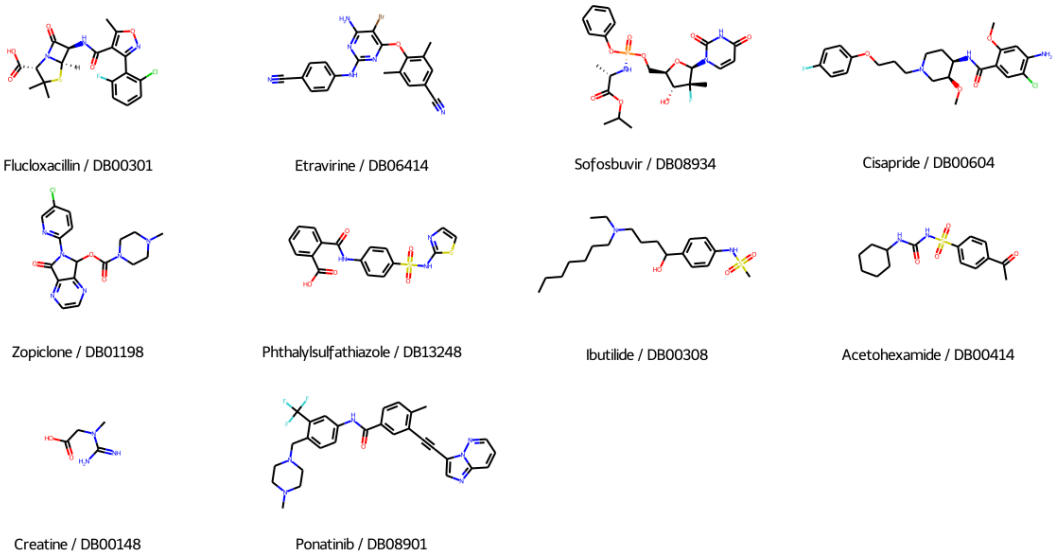
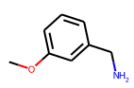


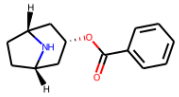
Figure B3: 10 drugs chosen out of 100 chemically diverse compounds from DrugBank (Figure B2) to show representative results for 2D-DQN methods, which were selected based on drug's binding target.

Table B4: Number of reagents for each reaction from Reaxys including seed, which was used for nc3D-DQN methods with the Malhotra set (Malhotra and Karanicolas, 2017). Note that the Suzuki coupling reaction is omitted to avoid boron-containing products.

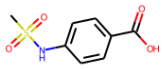
Reactions	Varied Synthon	All Reaxys Reagents
Amides	Acids	2202
	Amines	10079
N-Arylation	Amines	7211
	Halides	1555
S_NAr	Amines	7130
	Aromatic Halides	8444
Reductive Amination	Ketones/ Aldehydes	2096
	Amines	2356
Ester Coupling	Acids	464
	Alcohols	6961
Ether Coupling	Alcohols	6324
	Halides	3645
Sonogashira	Alkynes	1682
	Aromatic Halides	3506
Urea	Amines	2877
	Isocyanates	495
Sulfonamide	Sulfonyl Chlorides	735
	Amines	2896



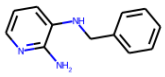
SZ4 / 3ati



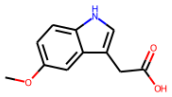
V63 / 2y54



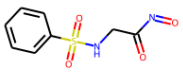
4MB / 2hds



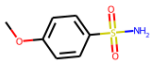
8AP / 2ohm



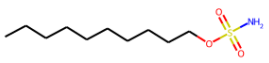
MYI / 3adu



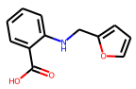
HS7 / 3f1a



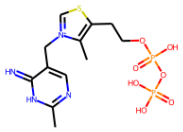
M4S / 3lka



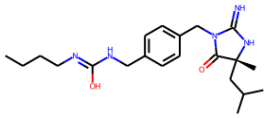
O48 / 3ibu



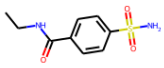
FUZ / 3rf5



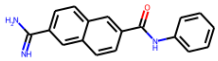
TPP / 3iaf



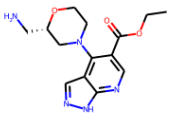
BDV / 3l5d



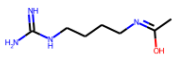
RYV / 3ryv



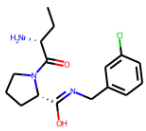
675 / 1owe



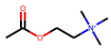
YM3 / 2ym3



G0R / 3bg8



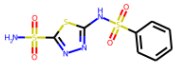
19U / 2zfp



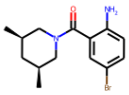
ACH / 2ha4



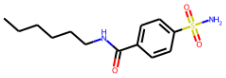
528 / 2piy



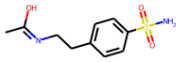
D8W / 3d8w



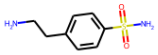
SX3 / 3cj2



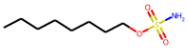
RZ8 / 3rz8



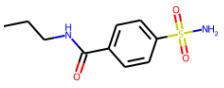
M25 / 2nns



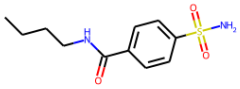
ZYX / 2nng



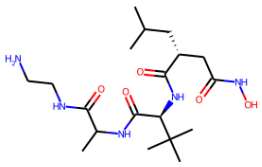
BOW / 3ibi



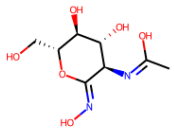
RYY / 3ryy



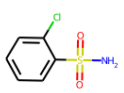
RZ0 / 3rz0



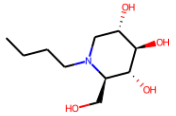
INN / 3kme



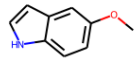
LOG / 2xm2



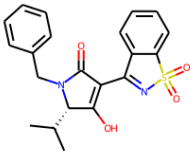
FB1 / 2weh



NBV / 2v3d



BZ3 / 3imc



B34 / 3d28

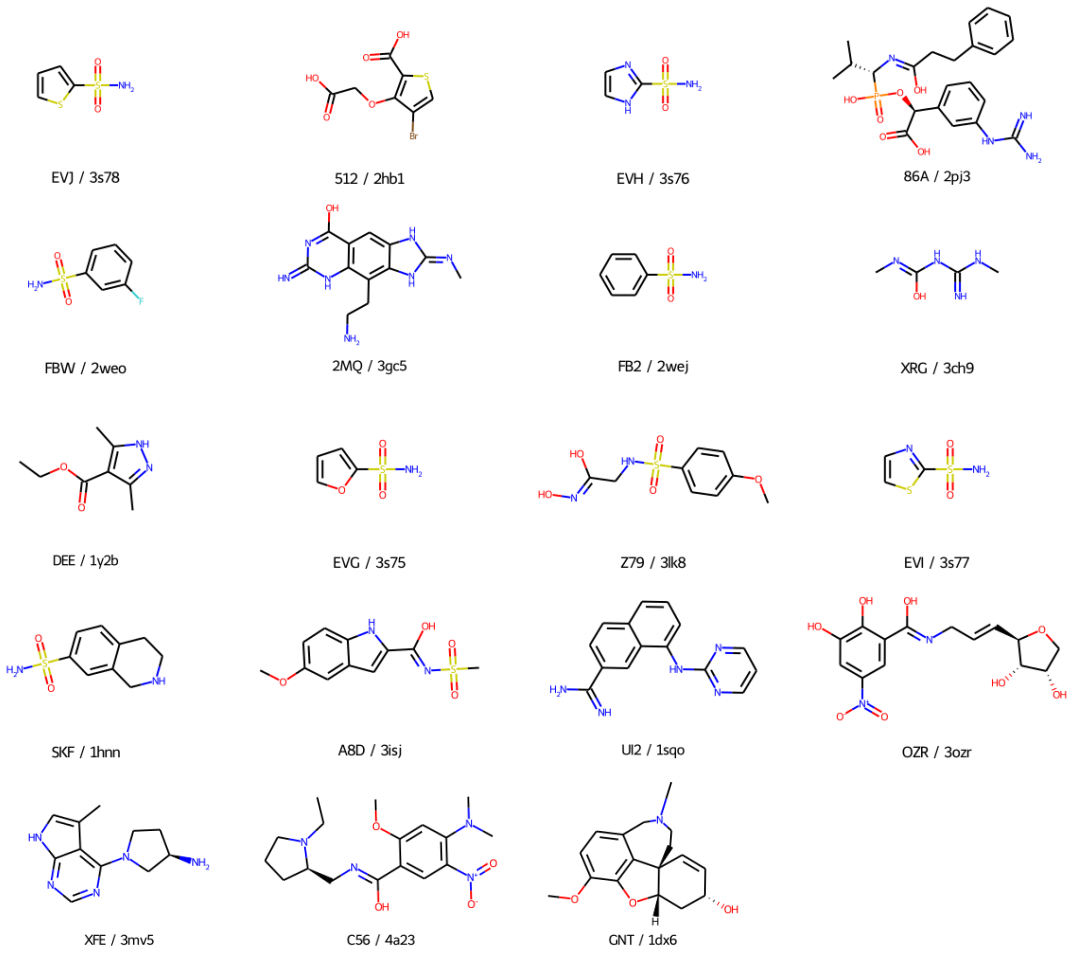


Figure B5: 51 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) used to test nc3D-MolDQN, nc3D-RxnDQN, and nc3D-RxnMolDQN (reward function of predicted ligand efficiency) with their receptors' and ligands' PDB ID.

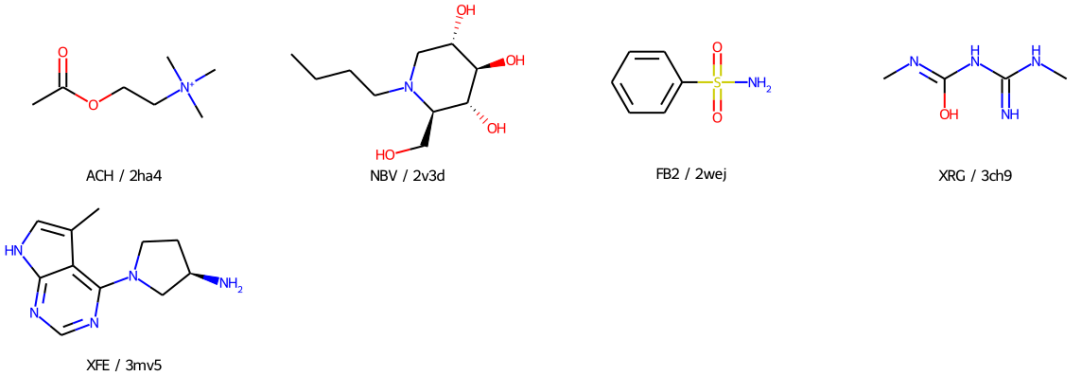


Figure B6: 5 Malhotra smaller ligands chosen from 51 Malhotra smaller ligands (Figure B5) to show representative results generated by nc3D-DQN (LE) methods. These ligands were chosen by computing for the chemical diversity of the smaller ligand (Figure B5) using RDKit (Landrum, 2019) .

Table B7: Number of reagents for each reaction from Reaxys without seed, which was used in c3D-DQN methods with SARS-CoV-2 main protease (Douangamath *et al.*, 2020). Note that the Suzuki coupling reaction is omitted to avoid boron-containing products.

Reactions	**Varied Synthon**	**All Reaxys Reagents**
Amides	Acids	2190
	Amines	10060
N-Arylation	Amines	7098
	Halides	1510
S_NAr	Amines	7046
	Aromatic Halides	8447
Reductive Amination	Ketones/ Aldehydes	2040
	Amines	3032
Ester Coupling	Acids	459
	Alcohols	6948
Ether Coupling	Alcohols	6290
	Halides	3614
Sonogashira	Alkynes	1682
	Aromatic Halides	3506
Urea	Amines	2877
	Isocyanates	495
Sulfonamide	Sulfonyl Chlorides	708
	Amines	2883

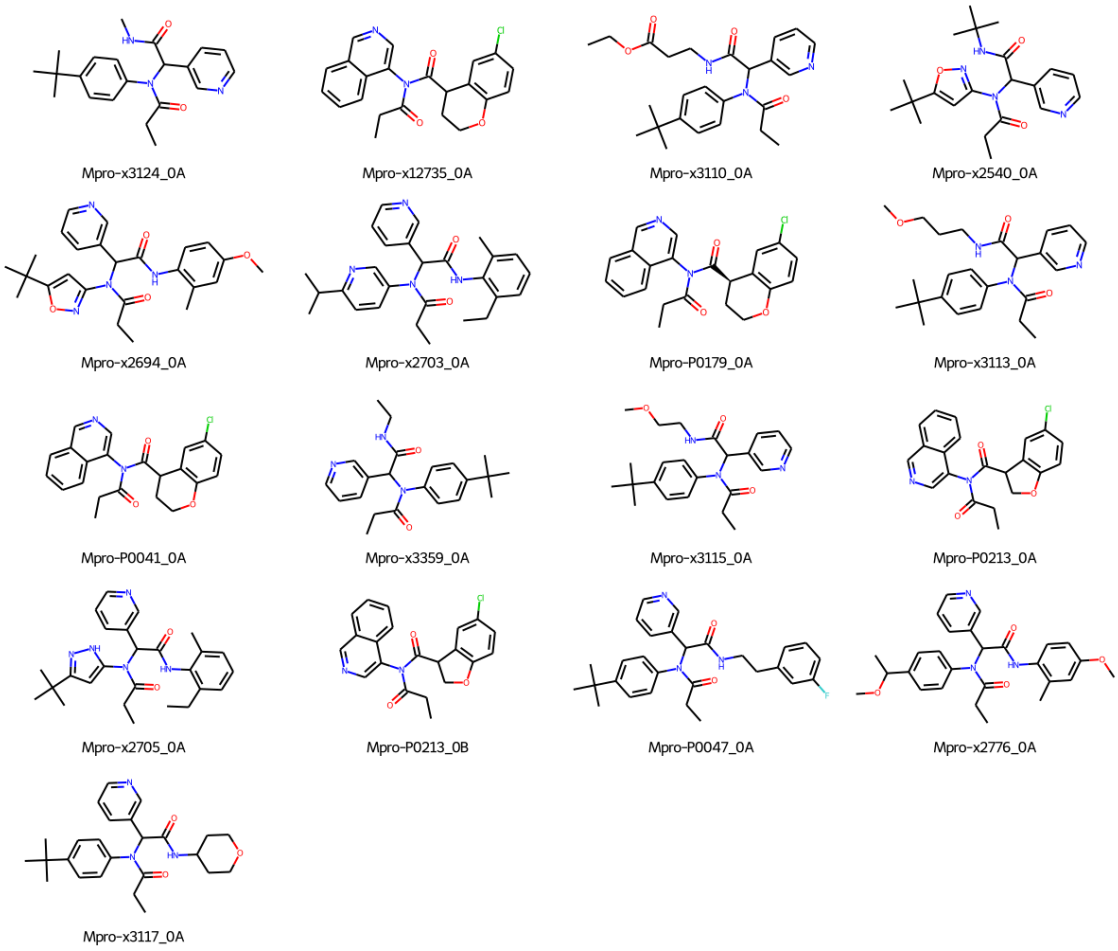
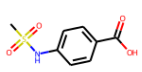


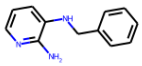
Figure B8: 17 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) used to test c3D-MolDQN, c3D-RxnDQN, and c3D-RxnMolDQN (reward function of predicted ligand efficiency).



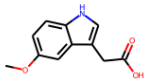
Figure B9: 3 covalent ligands from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) used to show representative results generated by c3D-DQN methods. These ligands were chosen by computing for the chemical diversity of the 17 covalent ligands (Figure B8) using RDKit (Landrum, 2019) .



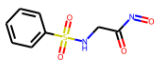
4MB / 2hds



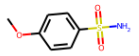
8AP / 2ohm



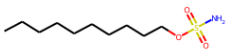
MYI / 3adu



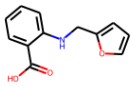
HS7 / 3f1a



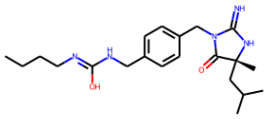
M4S / 3lka



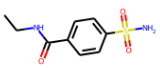
O48 / 3ibu



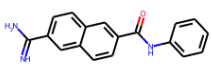
FUZ / 3rf5



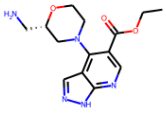
BDV / 3l5d



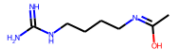
RYV / 3ryv



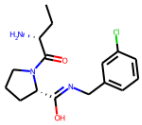
675 / 1owe



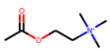
YM3 / 2ym3



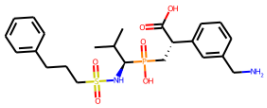
G0R / 3bg8



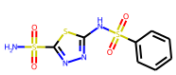
19U / 2zfp



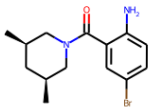
ACH / 2ha4



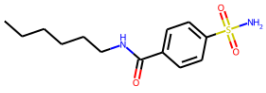
528 / 2piy



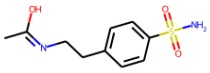
D8W / 3d8w



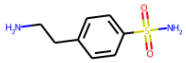
SX3 / 3cj2



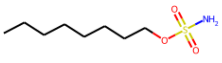
RZ8 / 3rz8



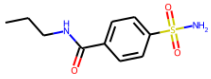
M25 / 2nns



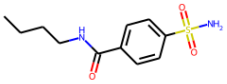
ZYX / 2nng



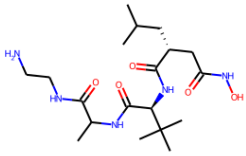
BOW / 3ibi



RYY / 3ryy



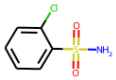
RZ0 / 3rz0



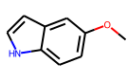
INN / 3kme



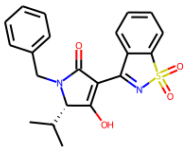
LOG / 2xm2



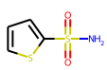
FB1 / 2weh



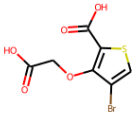
BZ3 / 3imc



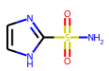
B34 / 3d28



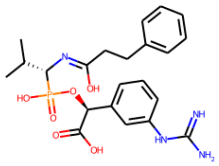
EVJ / 3s78



512 / 2hb1



EVH / 3s76



86A / 2pj3

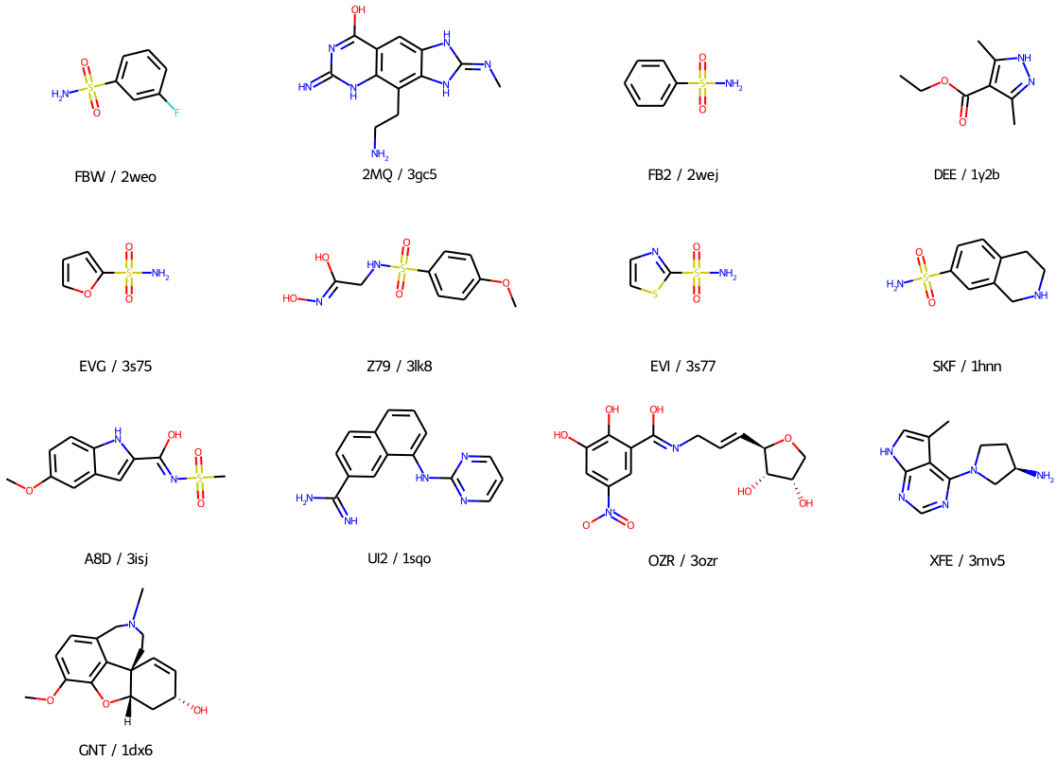


Figure B10: 45 Malhotra smaller ligands (Malhotra and Karanicolas, 2017) used to test nc3D-MolDQN, nc3D-RxnDQN, and nc3D-RxnMolDQN (reward function of normalized SuCOS) with their receptors' and ligands' PDB ID.

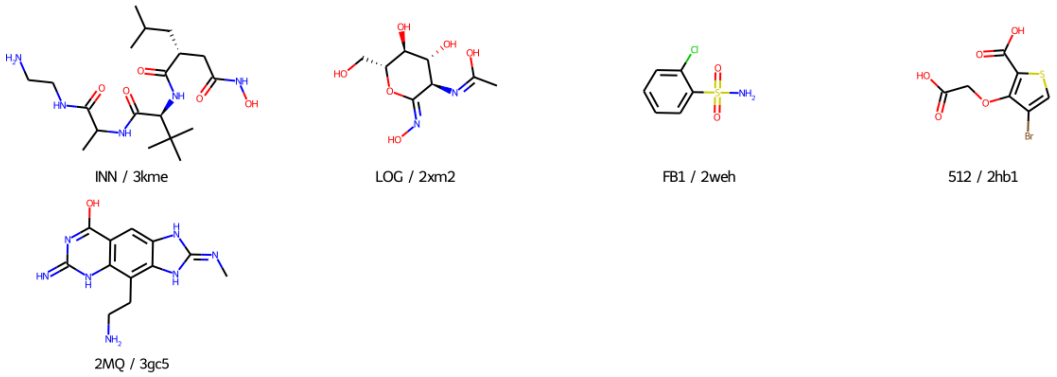


Figure B11: 5 Malhotra smaller ligands chosen from 45 Malhotra smaller ligands (Figure B10) to show representative results generated by nc3D-DQN (nSuCOS) methods. These ligands were chosen by computing for the chemical diversity of the smaller ligand using RDKit (Landrum, 2019) .

Appendix C: Chemical Space Coverage

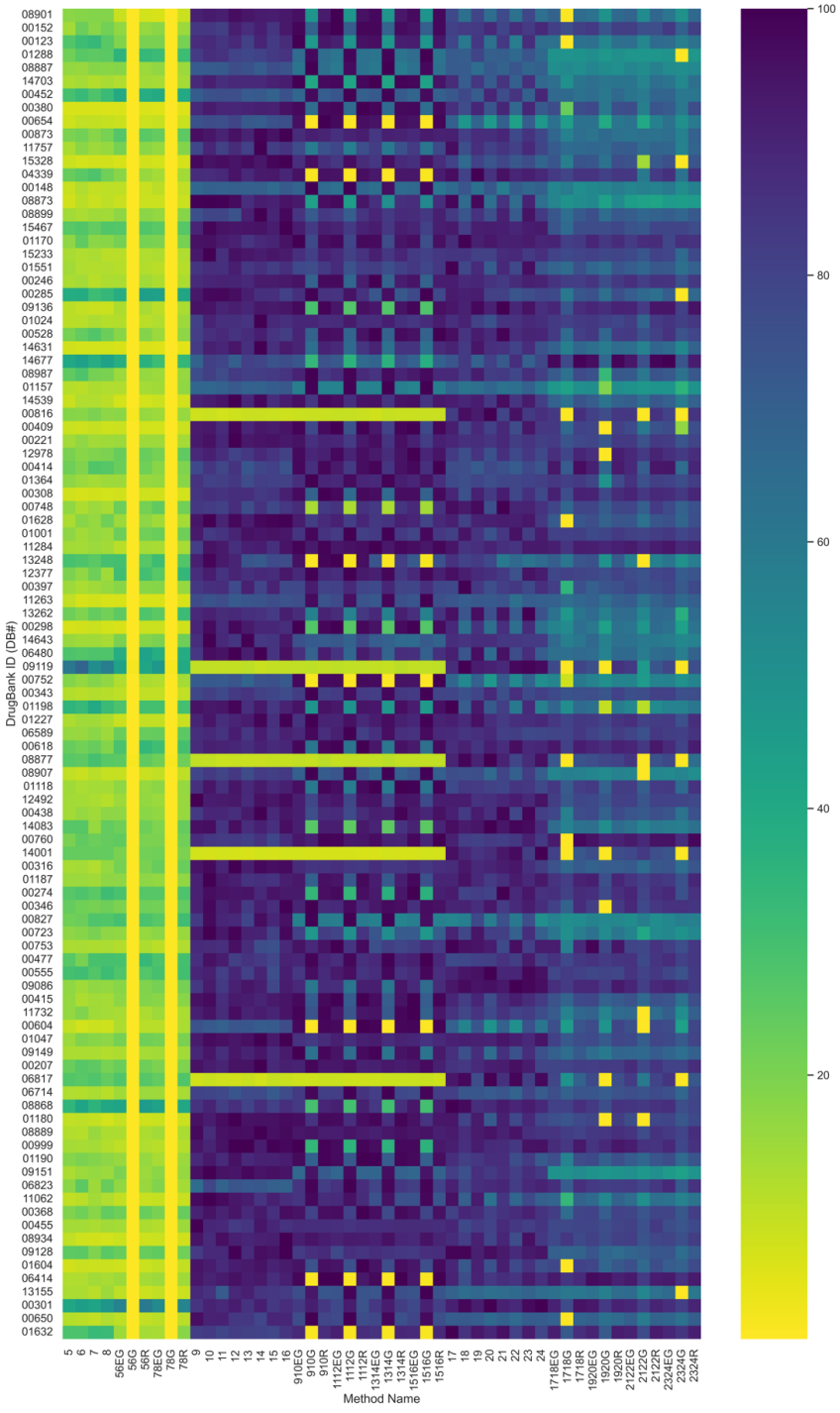


Figure C1: Per-drug relative percentage chemical space coverage for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) with its corresponding baselines computed using UMAP for 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED. “E”, “EG”, and “RW” stand for epsilon, epsilon greedy, and random walk, respectively.

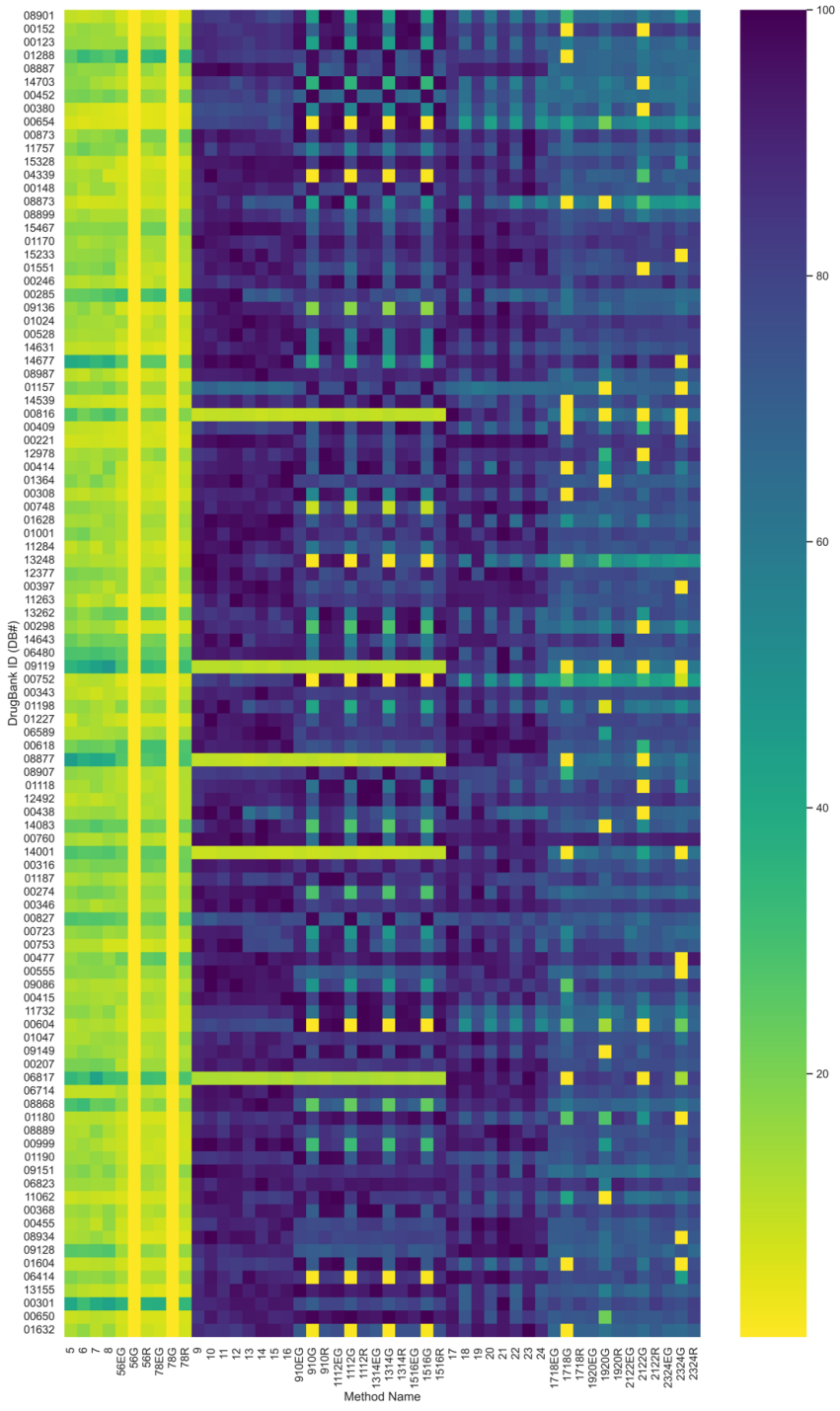


Figure C2: Per-drug relative percentage chemical space coverage for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) with its corresponding baselines computed using UMAP for 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, penalized logP. “E”, “EG”, and “RW” stand for epsilon, epsilon greedy, and random walk, respectively.

Appendix D: Validity

Table D1: Validity for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, QED. The values are approximated to three decimal places.

Method Number	Method	Baseline/DQN Methods	Average Validity for QED (%)
1	MolDQN	Naïve	100
2	MolDQN	Bootstrap	100
3	MolDQN	Naïve	100
4	MolDQN	Bootstrap	100
5	MolDQN	Naïve	100
6	MolDQN	Bootstrap	100
7	MolDQN	Naïve	100
8	MolDQN	Bootstrap	99.997±0.003
1/2	MolDQN	Epsilon Greedy	100
1/2	MolDQN	Greedy	100
1/2	MolDQN	Random Walk	100
3/4	MolDQN	Epsilon Greedy	100
3/4	MolDQN	Greedy	100
3/4	MolDQN	Random Walk	100
5/6	MolDQN	Epsilon Greedy	100
5/6	MolDQN	Greedy	100
5/6	MolDQN	Random Walk	100
7/8	MolDQN	Epsilon Greedy	100
7/8	MolDQN	Greedy	100
7/8	MolDQN	Random Walk	100
9	RxnDQN	Naïve	100
10	RxnDQN	Bootstrap	100
11	RxnDQN	Naïve	100
12	RxnDQN	Bootstrap	100
13	RxnDQN	Naïve	100
14	RxnDQN	Bootstrap	100
15	RxnDQN	Naïve	100
16	RxnDQN	Bootstrap	100
9/10	RxnDQN	Epsilon Greedy	100
9/10	RxnDQN	Greedy	100
9/10	RxnDQN	Random Walk	100
11/12	RxnDQN	Epsilon Greedy	100
11/12	RxnDQN	Greedy	100
11/12	RxnDQN	Random Walk	100
13/14	RxnDQN	Epsilon Greedy	100
13/14	RxnDQN	Greedy	100
13/14	RxnDQN	Random Walk	100
15/16	RxnDQN	Epsilon Greedy	100
15/16	RxnDQN	Greedy	100
15/16	RxnDQN	Random Walk	100
17	RxnMolDQN	Naive	100
18	RxnMolDQN	Bootstrap	100
19	RxnMolDQN	Naive	100
20	RxnMolDQN	Bootstrap	100
21	RxnMolDQN	Naive	100
22	RxnMolDQN	Bootstrap	99.997±0.003
23	RxnMolDQN	Naive	100
24	RxnMolDQN	Bootstrap	100
17/18	RxnMolDQN	Epsilon Greedy	100

17/18	RxnMolDQN	Greedy	100
17/18	RxnMolDQN	Random Walk	100
19/20	RxnMolDQN	Epsilon Greedy	100
19/20	RxnMolDQN	Greedy	100
19/20	RxnMolDQN	Random Walk	100
21/22	RxnMolDQN	Epsilon Greedy	100
21/22	RxnMolDQN	Greedy	100
21/22	RxnMolDQN	Random Walk	100
23/24	RxnMolDQN	Epsilon Greedy	100
23/24	RxnMolDQN	Greedy	100
23/24	RxnMolDQN	Random Walk	100

Table D2: Validity for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, penalized logP.

Method Number	Method	Baseline/DQN Methods	Average Validity for Penalized logP (%)
1	MolDQN	Naïve	100
2	MolDQN	Bootstrap	100
3	MolDQN	Naïve	100
4	MolDQN	Bootstrap	100
5	MolDQN	Naïve	100
6	MolDQN	Bootstrap	100
7	MolDQN	Naïve	100
8	MolDQN	Bootstrap	100
1/2	MolDQN	Epsilon Greedy	100
1/2	MolDQN	Greedy	100
1/2	MolDQN	Random Walk	100
3/4	MolDQN	Epsilon Greedy	100
3/4	MolDQN	Greedy	100
3/4	MolDQN	Random Walk	100
5/6	MolDQN	Epsilon Greedy	100
5/6	MolDQN	Greedy	100
5/6	MolDQN	Random Walk	100
7/8	MolDQN	Epsilon Greedy	100
7/8	MolDQN	Greedy	100
7/8	MolDQN	Random Walk	100
9	RxnDQN	Naïve	100
10	RxnDQN	Bootstrap	100
11	RxnDQN	Naïve	100
12	RxnDQN	Bootstrap	100
13	RxnDQN	Naïve	100
14	RxnDQN	Bootstrap	100
15	RxnDQN	Naïve	100
16	RxnDQN	Bootstrap	100
9/10	RxnDQN	Epsilon Greedy	100
9/10	RxnDQN	Greedy	100
9/10	RxnDQN	Random Walk	100
11/12	RxnDQN	Epsilon Greedy	100
11/12	RxnDQN	Greedy	100
11/12	RxnDQN	Random Walk	100
13/14	RxnDQN	Epsilon Greedy	100
13/14	RxnDQN	Greedy	100
13/14	RxnDQN	Random Walk	100
15/16	RxnDQN	Epsilon Greedy	100
15/16	RxnDQN	Greedy	100
15/16	RxnDQN	Random Walk	100
17	RxnMolDQN	Naive	100
18	RxnMolDQN	Bootstrap	100
19	RxnMolDQN	Naive	100
20	RxnMolDQN	Bootstrap	100
21	RxnMolDQN	Naive	100
22	RxnMolDQN	Bootstrap	100
23	RxnMolDQN	Naive	100
24	RxnMolDQN	Bootstrap	100
17/18	RxnMolDQN	Epsilon Greedy	100
17/18	RxnMolDQN	Greedy	100
17/18	RxnMolDQN	Random Walk	100

19/20	RxnMolDQN	Epsilon Greedy	100
19/20	RxnMolDQN	Greedy	100
19/20	RxnMolDQN	Random Walk	100
21/22	RxnMolDQN	Epsilon Greedy	100
21/22	RxnMolDQN	Greedy	100
21/22	RxnMolDQN	Random Walk	100
23/24	RxnMolDQN	Epsilon Greedy	100
23/24	RxnMolDQN	Greedy	100
23/24	RxnMolDQN	Random Walk	100

Appendix E: Synthetic Accessibility

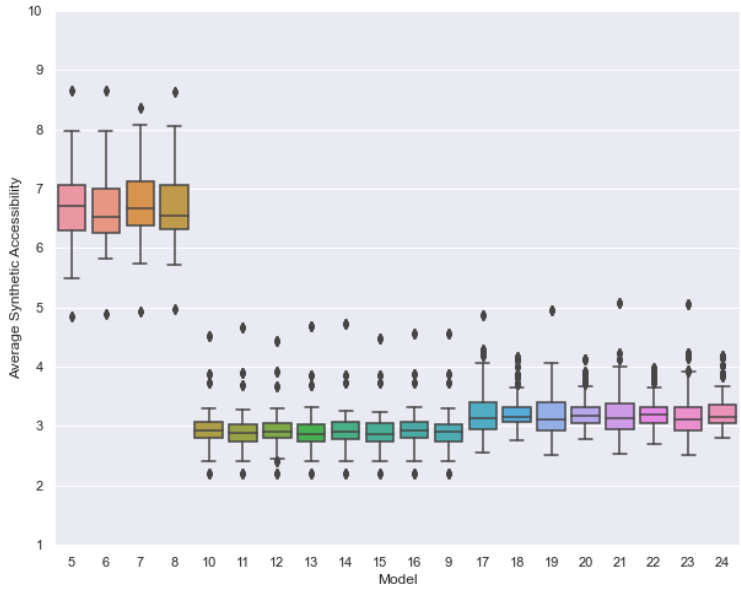


Figure E1: Average synthetic accessibility for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED.

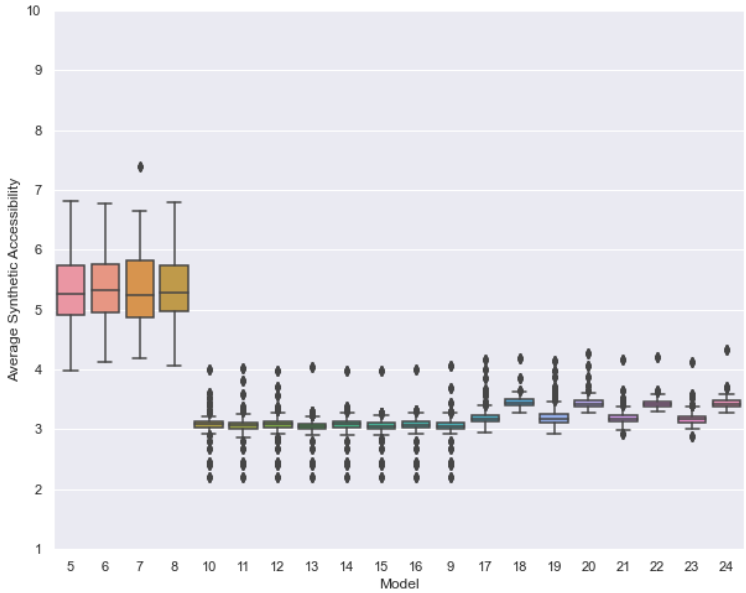


Figure E2: Average synthetic accessibility for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, penalized logP.

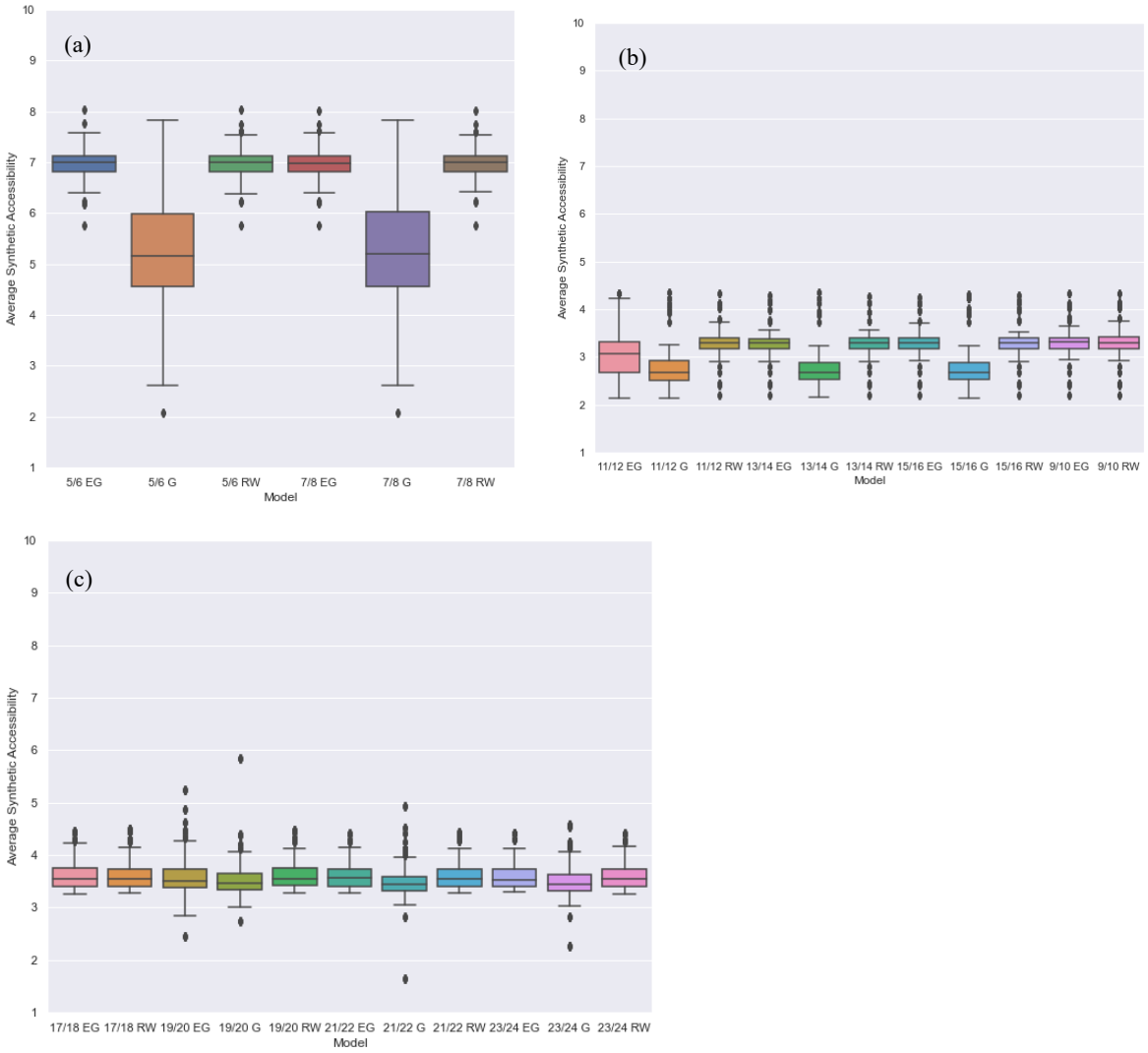


Figure E3: Average synthetic accessibility for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) baselines computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED. "E", "EG", and "RW" stand for epsilon, epsilon greedy, and random walk, respectively.

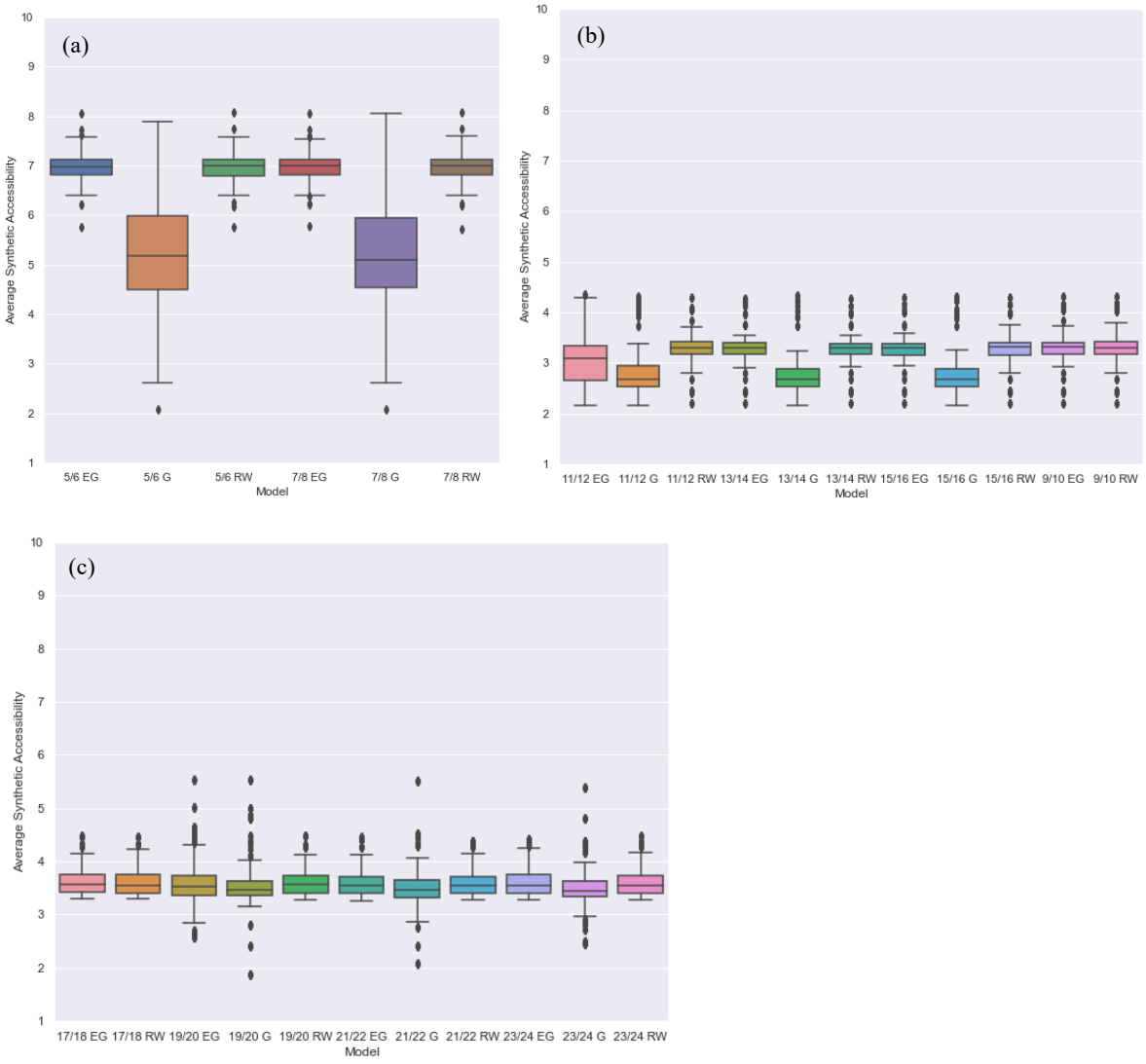


Figure E4: Average synthetic accessibility for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) baselines computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, penalized logP. "E", "EG", and "RW" stand for epsilon, epsilon greedy, and random walk, respectively.

Appendix F: Reward

Table F1: QED for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, QED. The values are approximated to three decimal places. Average of Best QED computes for the average of best QED per drug across 100 DrugBank (Wishart *et al.*, 2018) compounds (similar for Average of 2nd Best QED and Average of 3rd Best QED). Overall average QED calculates for the average QED for all molecules generated across 100 DrugBank (Wishart *et al.*, 2018) compounds.

Method Number	Method	DQN Method	Best QED	Average of Best QED	2nd Best QED	Average of 2nd Best QED	3rd Best QED	Average of 3rd Best QED1	Overall Average QED
1	MolDQN	Naive	0.932	-	0.929	-	0.924	-	0.525±0.245
2	MolDQN	Bootstrap	0.940	-	0.930	-	0.928	-	0.518±0.229
3	MolDQN	Naive	0.935	-	0.933	-	0.920	-	0.515±0.242
4	MolDQN	Bootstrap	0.929	-	0.925	-	0.923	-	0.516±0.227
5	MolDQN	Naive	0.947	0.849±0.125	0.947	0.844±0.126	0.947	0.841±0.126	0.418±0.275
6	MolDQN	Bootstrap	0.948	0.872±0.112	0.948	0.867±0.112	0.948	0.864±0.114	0.433±0.271
7	MolDQN	Naive	0.948	0.845±0.128	0.948	0.842±0.128	0.946	0.839±0.128	0.417±0.275
8	MolDQN	Bootstrap	0.948	0.868±0.118	0.948	0.861±0.121	0.948	0.858±0.121	0.427±0.270
1/2	MolDQN	Epsilon Greedy	0.808	-	0.693	-	0.690	-	0.170±0.126
1/2	MolDQN	Greedy	0.398	-	0.398	-	0.398	-	0.398
1/2	MolDQN	Random Walk	0.819	-	0.777	-	0.773	-	0.170±0.126
3/4	MolDQN	Epsilon Greedy	0.765	-	0.714	-	0.703	-	0.170±0.124
3/4	MolDQN	Greedy	0.398	-	0.398	-	0.398	-	0.398
3/4	MolDQN	Random Walk	0.793	-	0.767	-	0.762	-	0.171±0.127
5/6	MolDQN	Epsilon Greedy	0.920	0.520±0.210	0.857	0.473±0.204	0.841	0.445±0.195	0.063±0.073
5/6	MolDQN	Greedy	0.948	0.852±0.116	0.948	0.852±0.116	0.948	0.852±0.116	0.852±0.116
5/6	MolDQN	Random Walk	0.875	0.519±0.206	0.850	0.475±0.199	0.849	0.449±0.197	0.063±0.073
7/8	MolDQN	Epsilon Greedy	0.882	0.521±0.197	0.860	0.469±0.200	0.833	0.443±0.191	0.063±0.073
7/8	MolDQN	Greedy	0.948	0.851±0.118	0.948	0.851±0.118	0.948	0.851±0.118	0.851±0.118
7/8	MolDQN	Random Walk	0.888	0.522±0.202	0.842	0.473±0.196	0.838	0.448±0.194	0.063±0.073
9	RxnDQN	Naive	0.948	0.924±0.086	0.948	0.923±0.086	0.948	0.922±0.086	0.662±0.258
10	RxnDQN	Bootstrap	0.948	0.924±0.086	0.948	0.923±0.086	0.948	0.922±0.085	0.634±0.253
11	RxnDQN	Naive	0.948	0.925±0.086	0.948	0.923±0.086	0.948	0.922±0.086	0.663±0.257
12	RxnDQN	Bootstrap	0.948	0.924±0.086	0.948	0.923±0.086	0.948	0.921±0.086	0.636±0.253
13	RxnDQN	Naive	0.948	0.925±0.086	0.948	0.923±0.086	0.948	0.923±0.086	0.660±0.258
14	RxnDQN	Bootstrap	0.948	0.925±0.086	0.948	0.923±0.086	0.948	0.922±0.086	0.631±0.254
15	RxnDQN	Naive	0.948	0.924±0.086	0.948	0.923±0.086	0.948	0.923±0.086	0.659±0.258
16	RxnDQN	Bootstrap	0.948	0.925±0.086	0.948	0.923±0.086	0.948	0.922±0.086	0.633±0.254
9/10	RxnDQN	Epsilon Greedy	0.948	0.917±0.085	0.947	0.910±0.083	0.947	0.904±0.082	0.392±0.223
9/10	RxnDQN	Greedy	0.948	0.889±0.163	0.948	0.889±0.163	0.948	0.889±0.163	0.818±0.189
9/10	RxnDQN	Random Walk	0.948	0.918±0.085	0.947	0.911±0.084	0.947	0.906±0.083	0.392±0.223
11/12	RxnDQN	Epsilon Greedy	0.948	0.918±0.085	0.948	0.910±0.083	0.947	0.904±0.082	0.392±0.223
11/12	RxnDQN	Greedy	0.948	0.889±0.163	0.948	0.889±0.163	0.948	0.889±0.163	0.818±0.190

11/12	RxnDQN	Random Walk	0.948	0.917±0.085	0.948	0.910±0.083	0.947	0.904±0.082	0.392±0.223
13/14	RxnDQN	Epsilon Greedy	0.948	0.918±0.085	0.948	0.912±0.084	0.948	0.907±0.083	0.392±0.223
13/14	RxnDQN	Greedy	0.948	0.890±0.163	0.948	0.889±0.163	0.948	0.889±0.163	0.819±0.189
13/14	RxnDQN	Random Walk	0.948	0.918±0.085	0.948	0.911±0.083	0.947	0.905±0.082	0.392±0.223
15/16	RxnDQN	Epsilon Greedy	0.948	0.917±0.085	0.947	0.910±0.083	0.947	0.905±0.082	0.392±0.223
15/16	RxnDQN	Greedy	0.948	0.889±0.163	0.948	0.889±0.163	0.948	0.889±0.163	0.819±0.189
15/16	RxnDQN	Random Walk	0.948	0.917±0.085	0.948	0.909±0.083	0.947	0.904±0.082	0.391±0.223
17	RxnMolDQN	Naive	0.948	0.944±0.004	0.948	0.943±0.004	0.948	0.943±0.005	0.674±0.274
18	RxnMolDQN	Bootstrap	0.948	0.946±0.002	0.948	0.946±0.002	0.948	0.945±0.002	0.638±0.275
19	RxnMolDQN	Naive	0.948	0.944±0.004	0.948	0.943±0.004	0.948	0.943±0.005	0.671±0.274
20	RxnMolDQN	Bootstrap	0.948	0.946±0.002	0.948	0.946±0.002	0.948	0.945±0.002	0.637±0.273
21	RxnMolDQN	Naive	0.948	0.943±0.005	0.948	0.942±0.006	0.948	0.942±0.007	0.671±0.273
22	RxnMolDQN	Bootstrap	0.948	0.946±0.002	0.948	0.946±0.002	0.948	0.945±0.002	0.636±0.273
23	RxnMolDQN	Naive	0.948	0.944±0.005	0.948	0.943±0.006	0.948	0.942±0.007	0.672±0.274
24	RxnMolDQN	Bootstrap	0.948	0.946±0.002	0.948	0.946±0.003	0.948	0.946±0.003	0.634±0.274
17/18	RxnMolDQN	Epsilon Greedy	0.946	0.938±0.006	0.944	0.930±0.009	0.940	0.924±0.010	0.357±0.219
17/18	RxnMolDQN	Greedy	0.948	0.899±0.110	0.948	0.892±0.108	0.948	0.885±0.106	0.397±0.229
17/18	RxnMolDQN	Random Walk	0.948	0.937±0.008	0.947	0.930±0.010	0.946	0.924±0.011	0.355±0.219
19/20	RxnMolDQN	Epsilon Greedy	0.948	0.937±0.007	0.947	0.929±0.009	0.947	0.923±0.010	0.356±0.219
19/20	RxnMolDQN	Greedy	0.948	0.909±0.099	0.946	0.900±0.098	0.946	0.892±0.098	0.378±0.222
19/20	RxnMolDQN	Random Walk	0.948	0.937±0.008	0.946	0.931±0.009	0.940	0.925±0.010	0.356±0.219
21/22	RxnMolDQN	Epsilon Greedy	0.948	0.938±0.007	0.945	0.930±0.009	0.944	0.922±0.011	0.356±0.219
21/22	RxnMolDQN	Greedy	0.948	0.900±0.142	0.948	0.890±0.140	0.946	0.883±0.139	0.373±0.218
21/22	RxnMolDQN	Random Walk	0.948	0.937±0.008	0.944	0.929±0.010	0.943	0.923±0.011	0.356±0.219
23/24	RxnMolDQN	Epsilon Greedy	0.948	0.937±0.007	0.947	0.930±0.009	0.941	0.923±0.010	0.356±0.219
23/24	RxnMolDQN	Greedy	0.947	0.896±0.140	0.946	0.891±0.138	0.945	0.885±0.137	0.378±0.221
23/24	RxnMolDQN	Random Walk	0.948	0.937±0.008	0.946	0.929±0.010	0.940	0.923±0.010	0.355±0.219

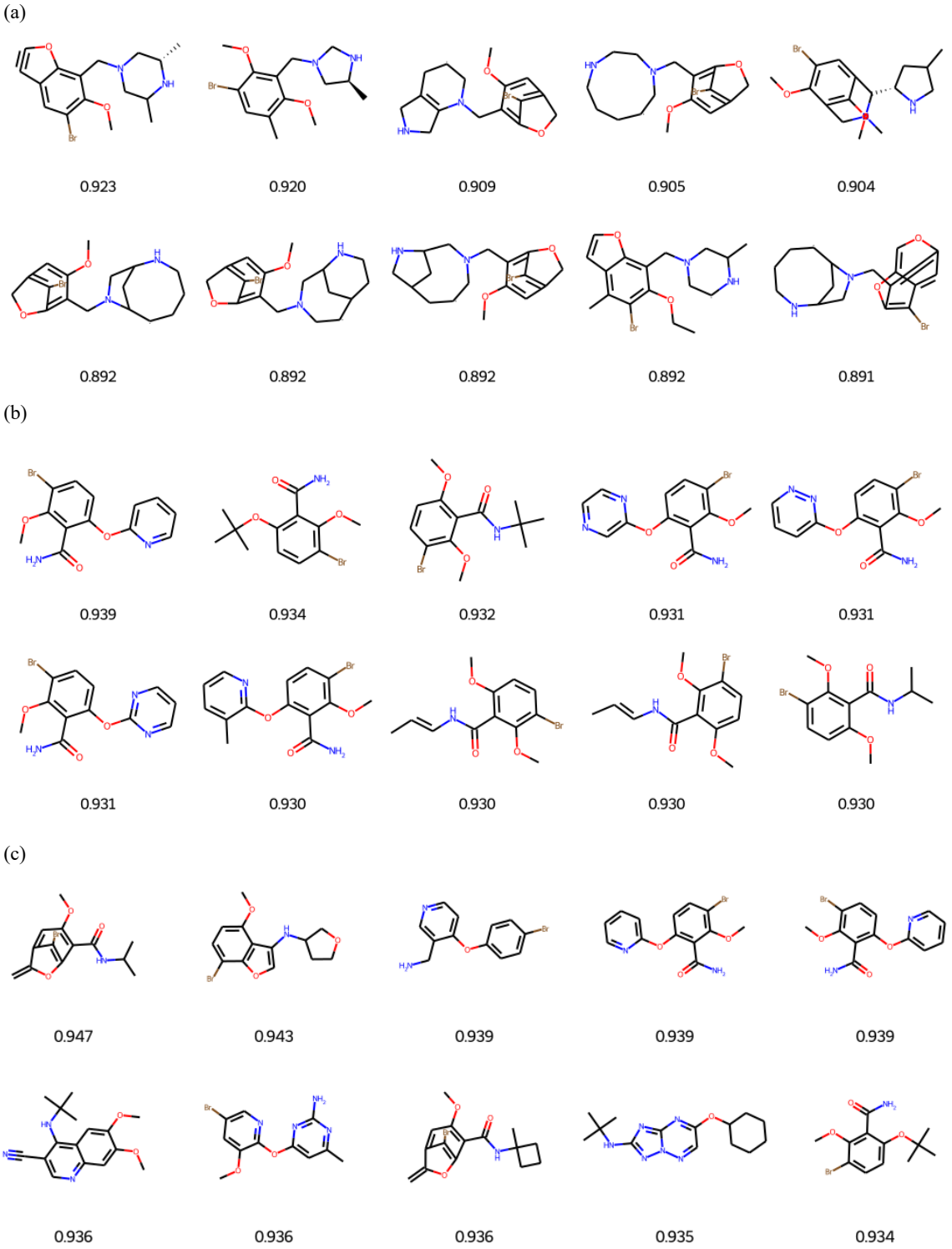


Figure F2: Top 10 QED scored compounds when using flucloxacillin/DB00301 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

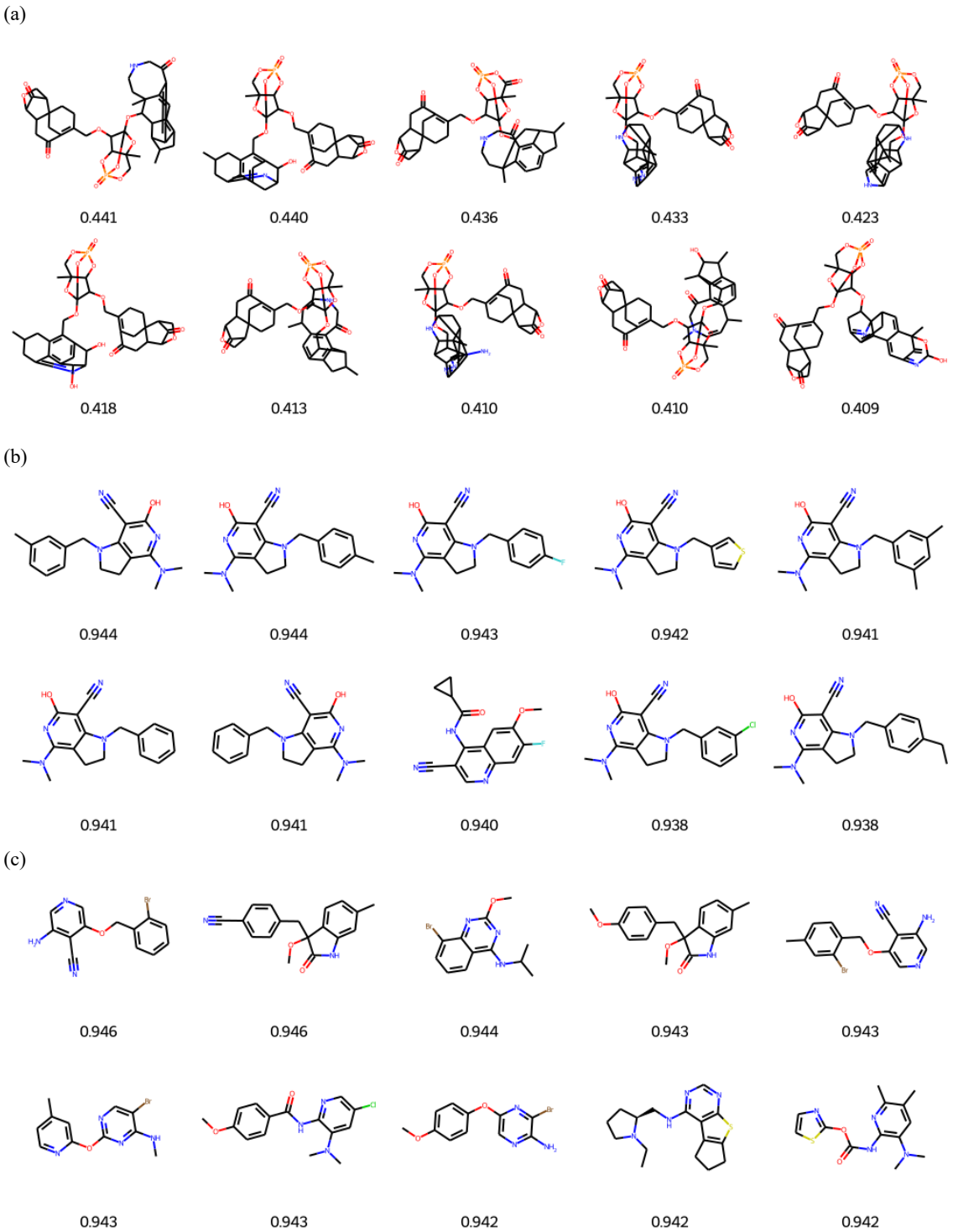
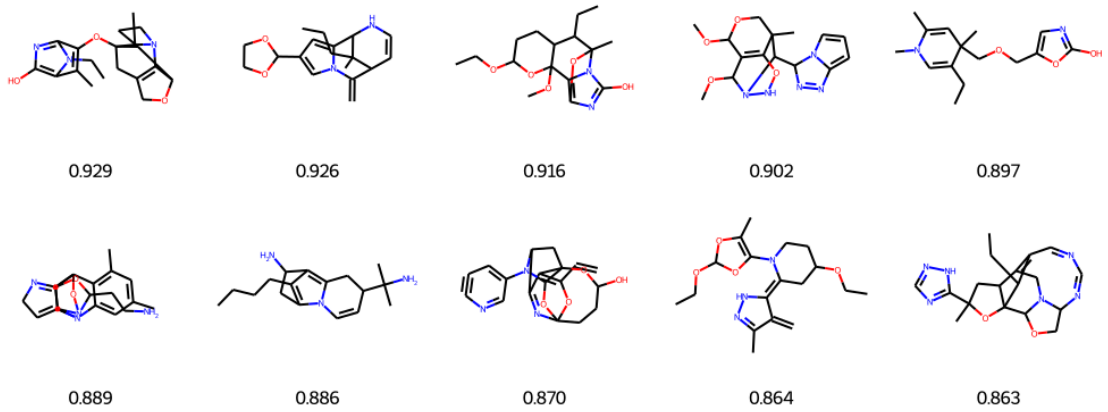
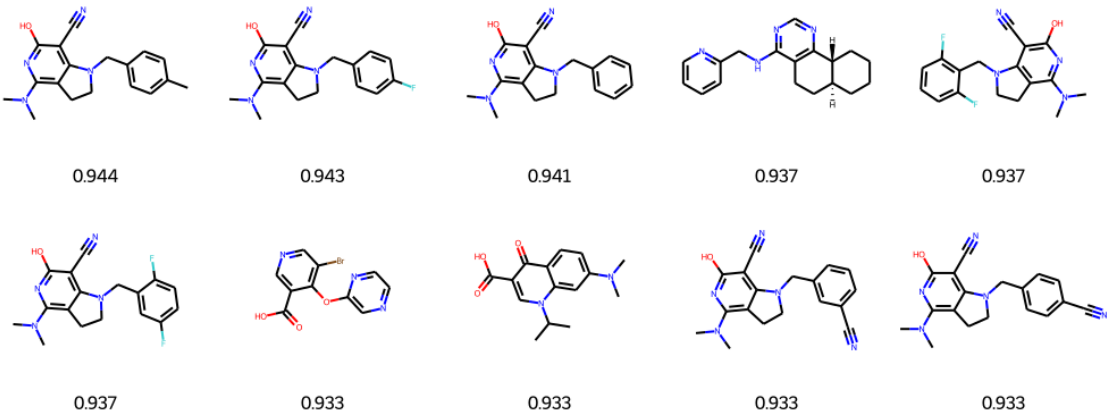


Figure F3: Top 10 QED scored compounds when using etravirine/DB06414 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

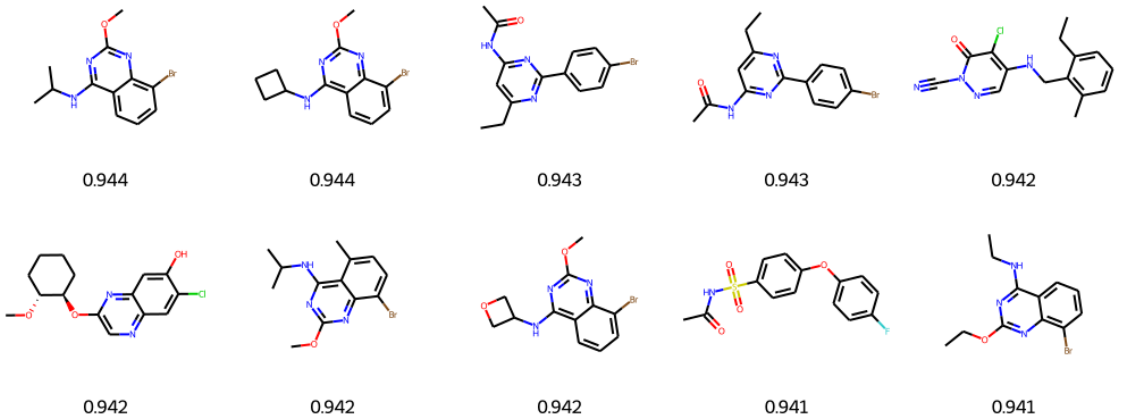
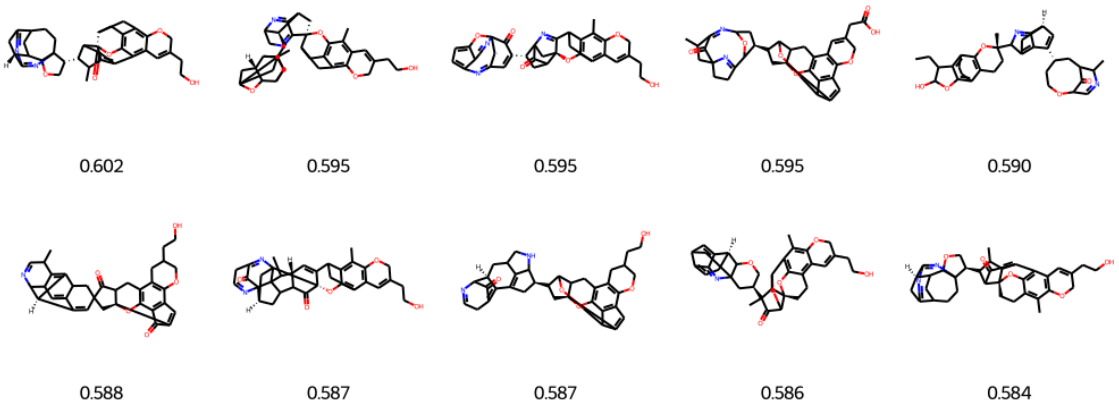
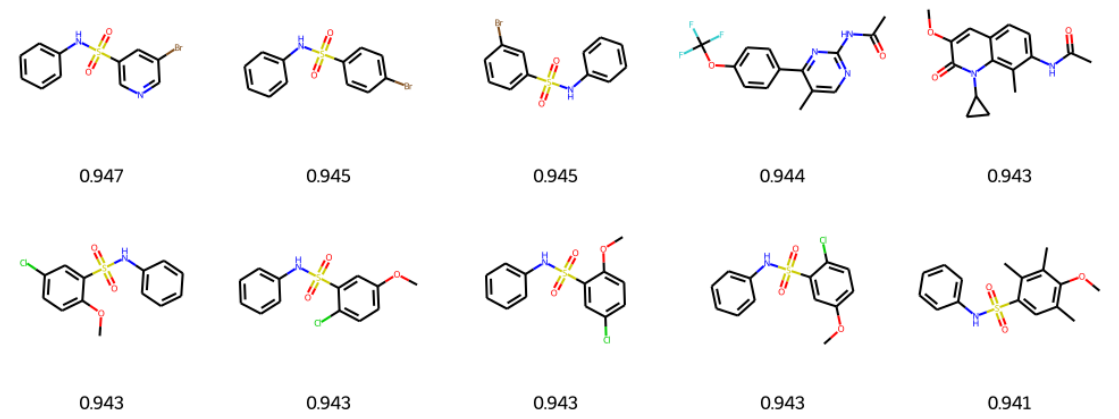


Figure F4: Top 10 QED scored compounds when using sofosbuvir/DB08934 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

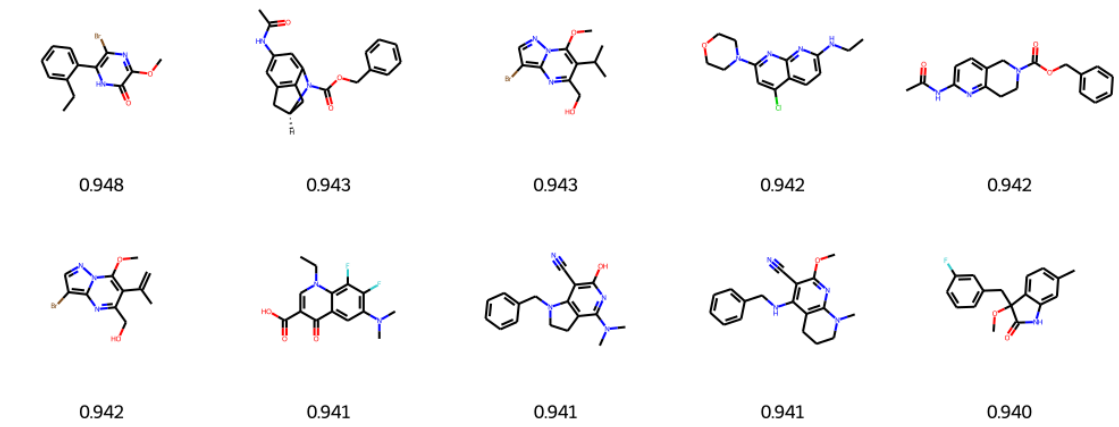
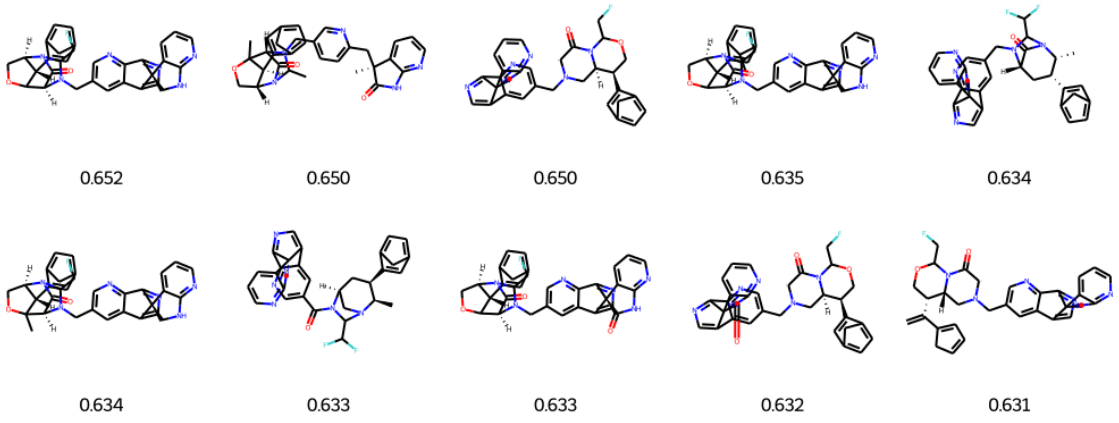
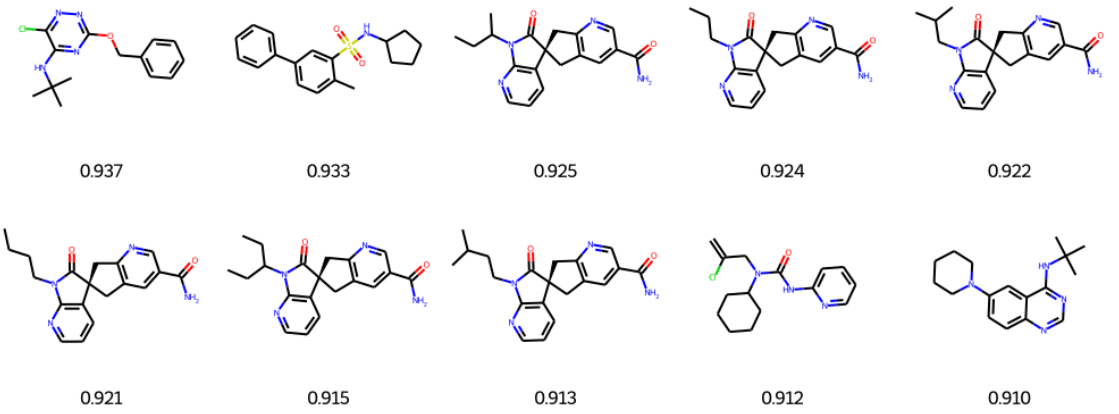


Figure F5: Top 10 QED scored compounds when using cisapride/DB00604 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

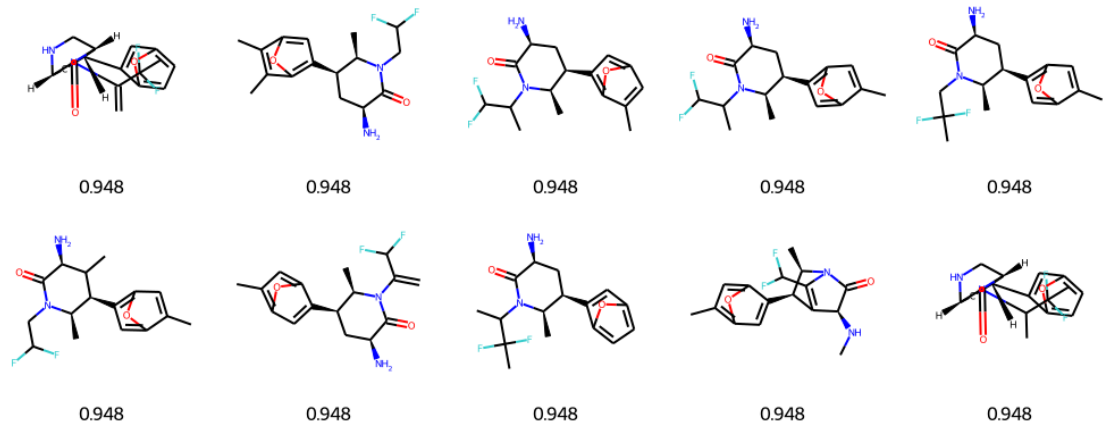
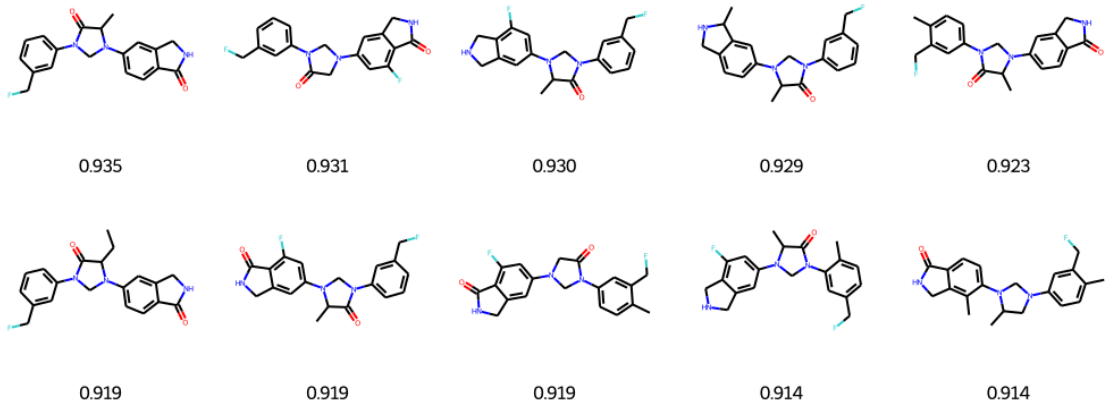
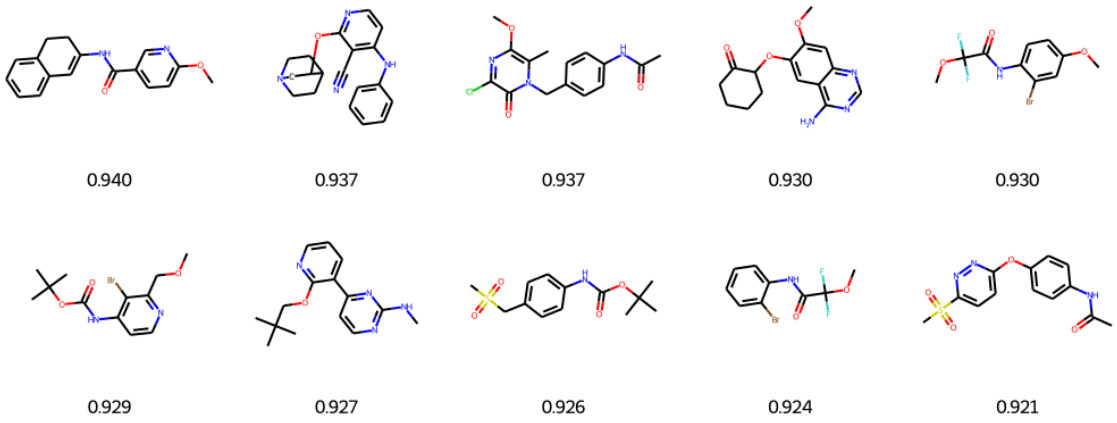


Figure F6: Top 10 QED scored compounds when using zopiclone/DB00198 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

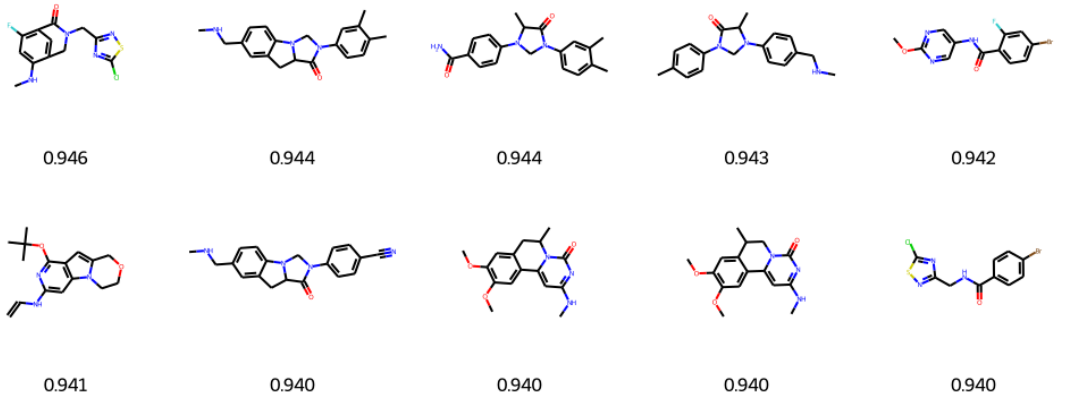
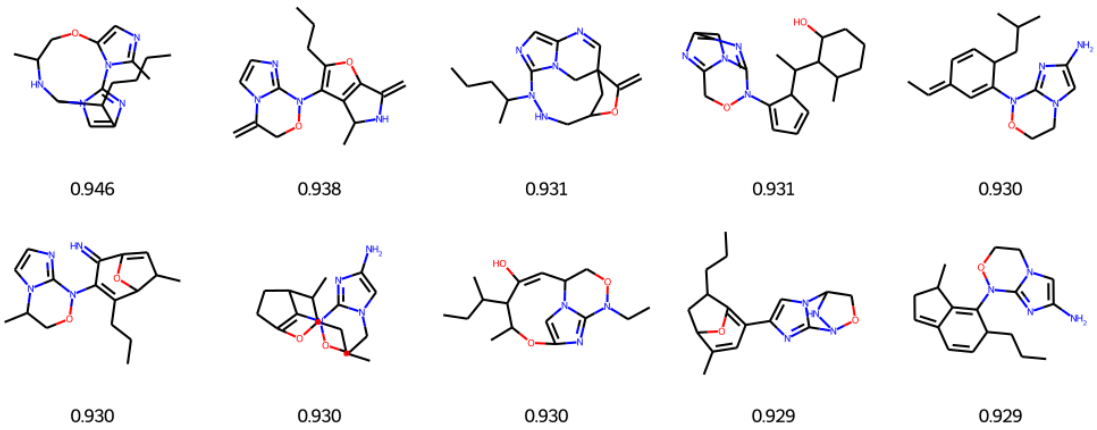
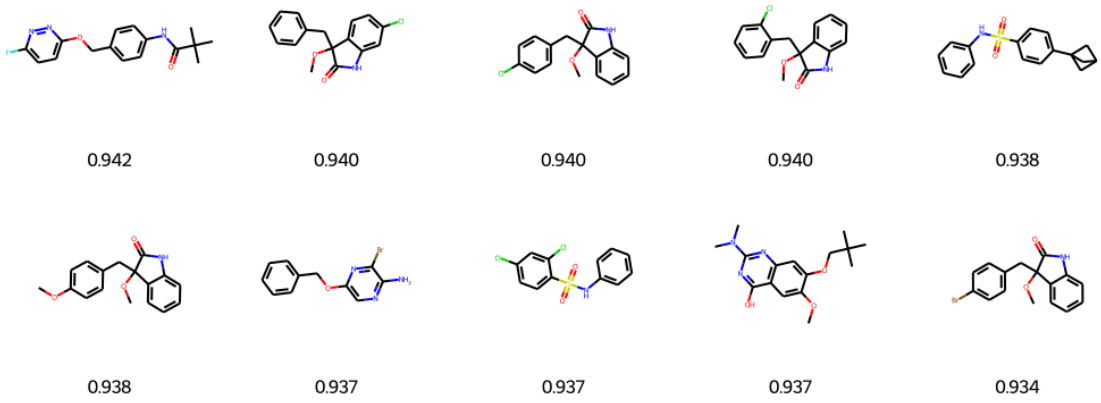


Figure F7: Top 10 QED scored compounds when using phthalylsulfathiazole/DB13248 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

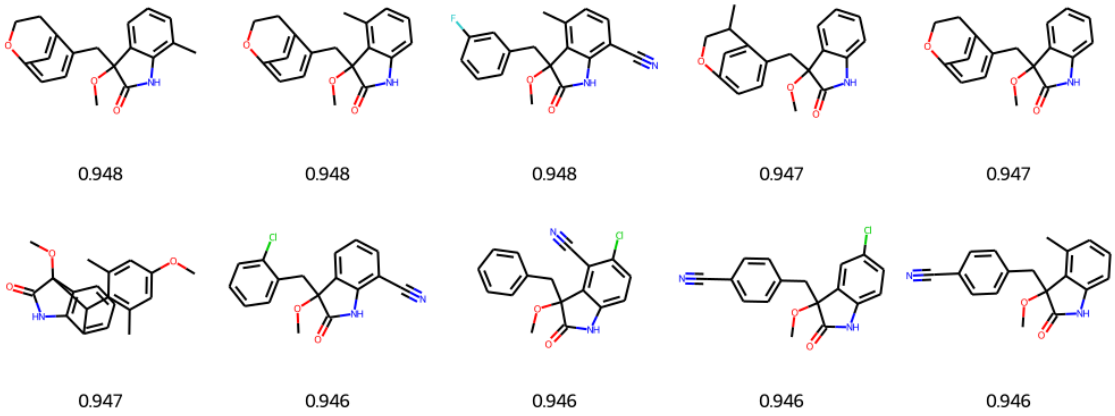
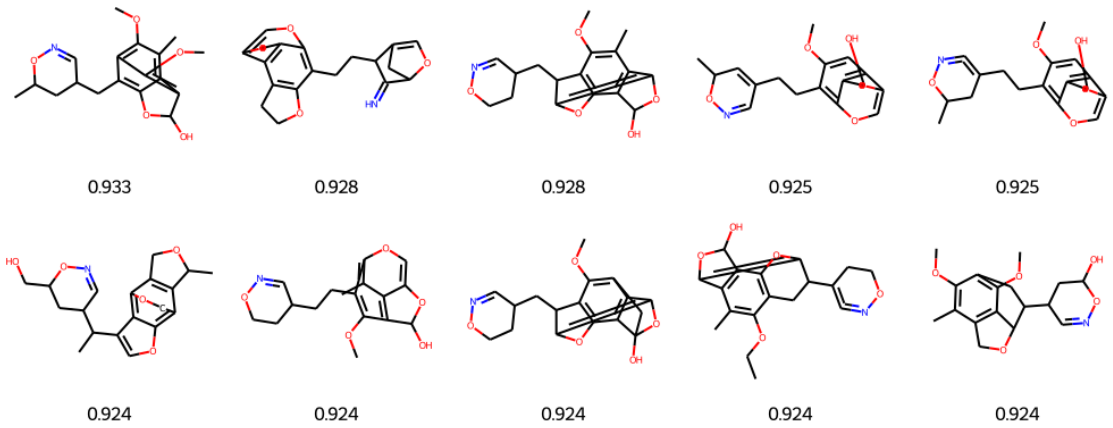
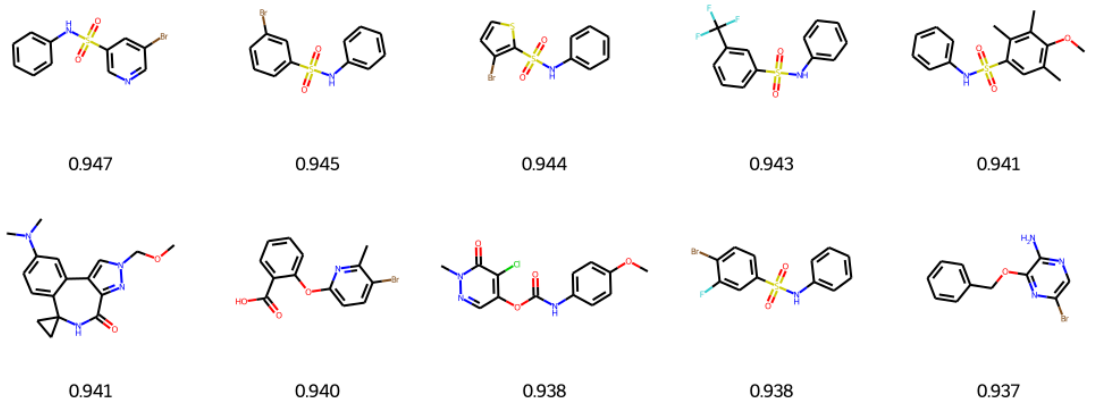


Figure F8: Top 10 QED scored compounds when using ibutilide/DB00308 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

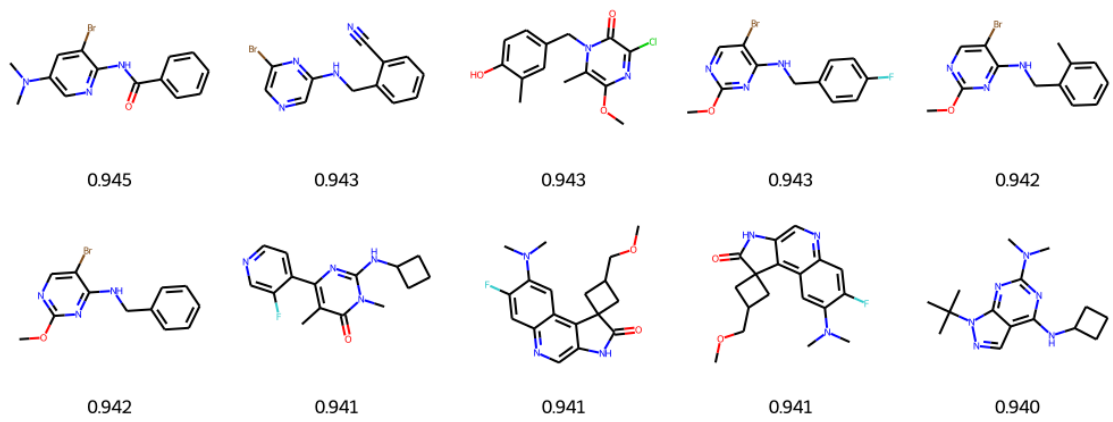
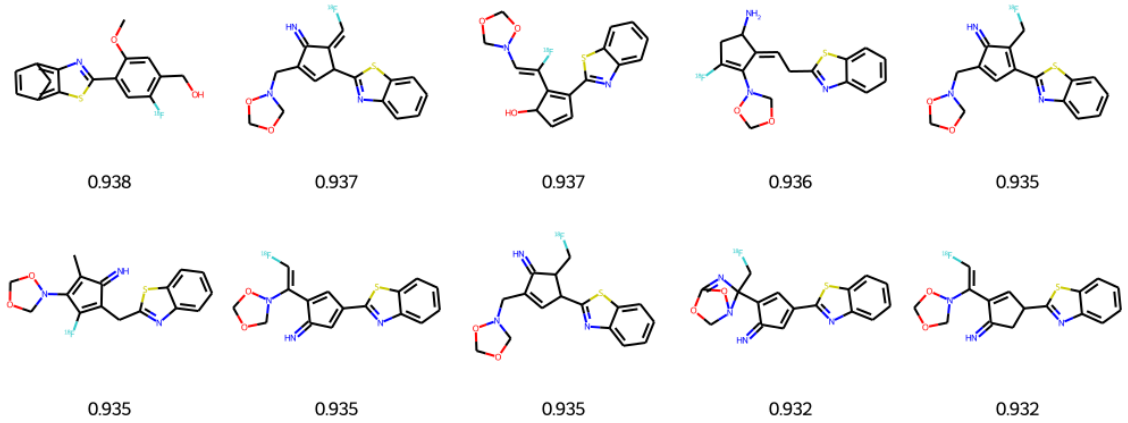
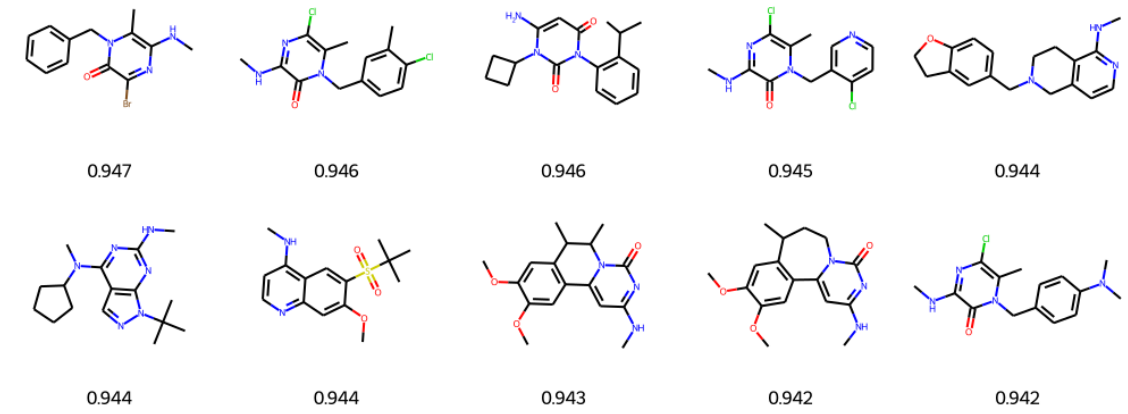


Figure F9: Top 10 QED scored compounds when using acetohexamide/DB00414 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

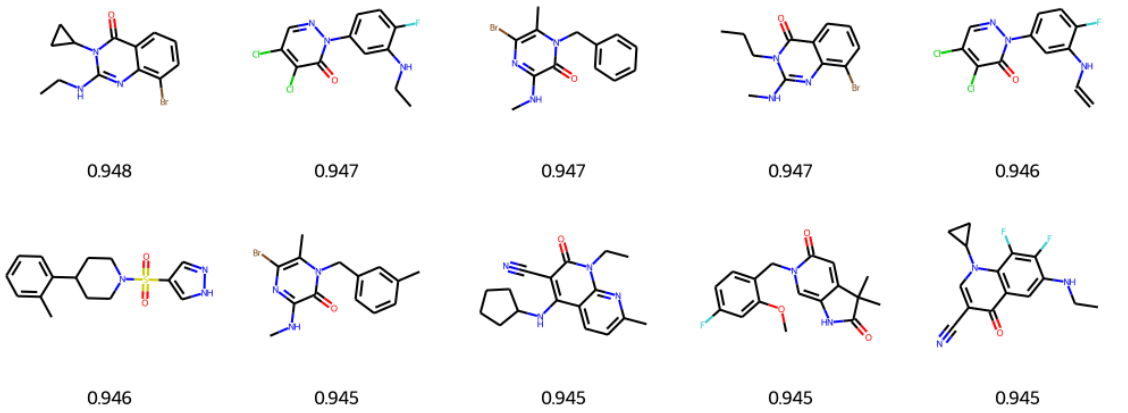
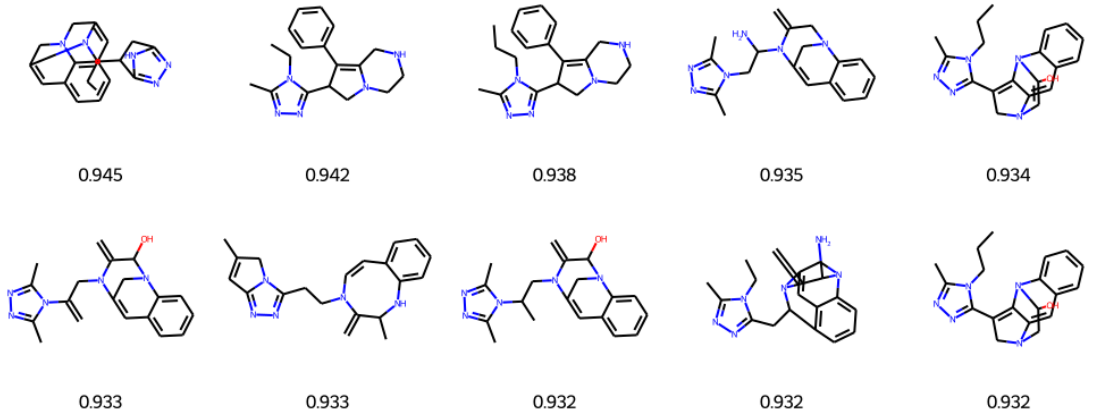
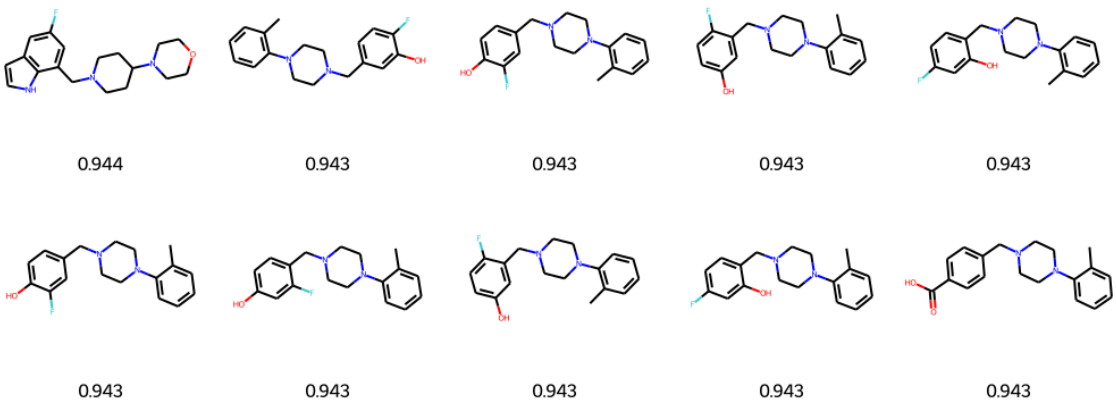


Figure F10: Top 10 QED scored compounds when using creatine/DB00148 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

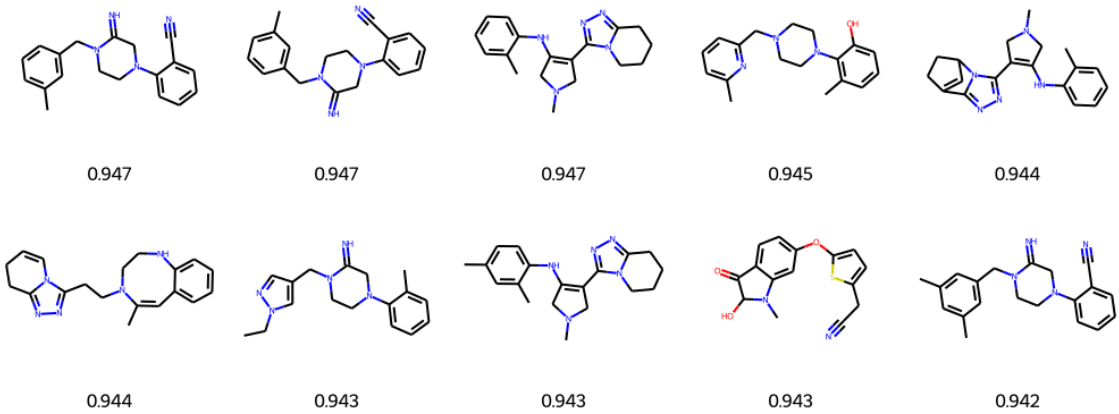


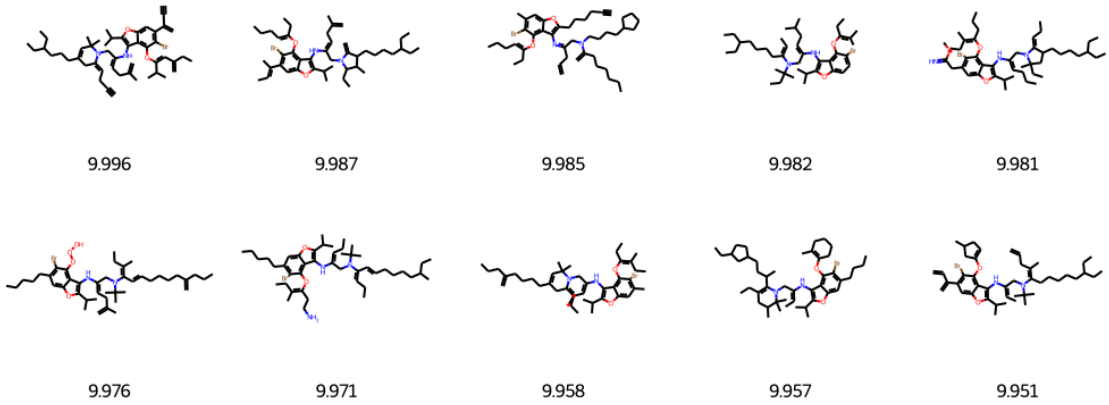
Figure F11: Top 10 QED scored compounds when using ponatinib/DB08901 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

Table F12: Penalized logP for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, penalized logP. The values are approximated to three decimal places. Average of Best p.logP computes for the average of best penalized logP per drug across 100 DrugBank (Wishart *et al.*, 2018) compounds (similar for Averages of 2nd Best p.logP and 3rd Best p.logP). Overall average penalized logP calculates for the average penalized logP for all molecules generated across 100 DrugBank (Wishart *et al.*, 2018) compounds.

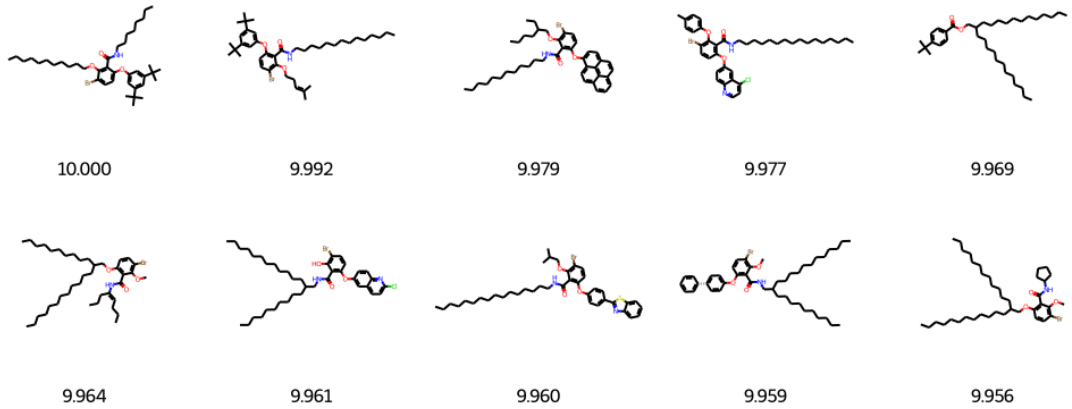
Method Number	Method	DQN Method	Best p.logP	Average of Best p.logP	2nd Best p.logP	Average of 2nd Best p.logP	3rd Best p.logP	Average of 3rd Best p.logP	Overall Average p.logP
1	MolDQN	Naive	9.997	-	9.997	-	9.997	-	2.437±5.929
2	MolDQN	Bootstrap	9.973	-	9.970	-	9.956	-	1.873±5.708
3	MolDQN	Naive	9.987	-	9.987	-	9.982	-	2.791±6.248
4	MolDQN	Bootstrap	9.991	-	9.987	-	9.976	-	2.079±6.021
5	MolDQN	Naive	10.000	9.943±0.409	9.999	9.935±0.413	9.999	9.930±0.413	4.227±6.503
6	MolDQN	Bootstrap	10.000	9.968±0.161	10.000	9.962±0.163	9.999	9.956±0.164	3.531±6.695
7	MolDQN	Naive	10.000	9.952±0.340	10.000	9.945±0.369	9.999	9.940±0.368	4.237±6.532
8	MolDQN	Bootstrap	10.000	9.958±0.183	10.000	9.949±0.185	10.000	9.944±0.185	3.476±6.646
1/2	MolDQN	Epsilon Greedy	-9.993	-	-9.987	-	-9.985	-	7.678±1.754
1/2	MolDQN	Greedy	-6.506	-	-6.506	-	-6.506	-	-6.506
1/2	MolDQN	Random Walk	-9.985	-	-9.984	-	-9.982	-	7.638±1.729
3/4	MolDQN	Epsilon Greedy	-9.999	-	-9.996	-	-9.988	-	7.633±1.645
3/4	MolDQN	Greedy	-6.506	-	-6.506	-	-6.506	-	-6.506
3/4	MolDQN	Random Walk	-9.987	-	-9.983	-	-9.980	-	7.715±1.698
5/6	MolDQN	Epsilon Greedy	4.434	3.581±5.745	3.677	5.845±5.442	2.894	6.875±5.014	7.456±2.763
5/6	MolDQN	Greedy	2.085	2.637±2.144	2.085	2.637±2.144	2.085	2.637±2.144	2.637±2.144
5/6	MolDQN	Random Walk	4.258	4.528±5.741	3.609	5.973±5.390	2.967	6.967±4.988	7.463±2.768
7/8	MolDQN	Epsilon Greedy	5.223	3.866±5.815	3.101	5.454±5.469	2.937	6.466±5.153	7.455±2.765
7/8	MolDQN	Greedy	2.085	2.533±1.998	2.085	2.533±1.998	2.085	2.533±1.998	2.533±1.998
7/8	MolDQN	Random Walk	3.594	3.835±5.722	3.591	5.459±5.470	2.896	-6.604±5.07	7.459±2.769
9	RxnDQN	Naive	10.000	9.421±2.528	9.999	9.416±2.527	9.999	9.412±2.526	5.977±4.804
10	RxnDQN	Bootstrap	10.000	9.420±2.528	10.000	9.416±2.527	9.999	9.411±2.526	5.794±4.785
11	RxnDQN	Naive	10.000	9.419±2.528	10.000	9.416±2.527	9.999	9.411±2.526	5.984±4.794
12	RxnDQN	Bootstrap	10.000	9.421±2.528	10.000	9.416±2.527	9.999	9.413±2.526	5.762±4.778
13	RxnDQN	Naive	10.000	9.421±2.528	10.000	9.416±2.527	9.999	9.412±2.526	5.960±4.785
14	RxnDQN	Bootstrap	10.000	9.421±2.528	9.999	9.417±2.527	9.998	9.413±2.526	5.765±4.762
15	RxnDQN	Naive	10.000	9.420±2.528	9.999	9.415±2.527	9.999	9.412±2.526	5.951±4.768
16	RxnDQN	Bootstrap	10.000	9.420±2.528	9.999	9.415±2.527	9.999	9.411±2.526	5.783±4.777
9/10	RxnDQN	Epsilon Greedy	9.994	8.843±2.440	9.957	8.439±2.532	9.895	8.221±2.322	0.844±2.557
9/10	RxnDQN	Greedy	4.978	2.206±1.409	4.978	2.217±1.312	4.978	2.149±1.301	0.037±1.594
9/10	RxnDQN	Random Walk	9.982	8.896±2.451	9.945	8.518±2.387	9.895	8.195±2.320	0.844±2.548
11/12	RxnDQN	Epsilon Greedy	9.998	8.875±2.447	9.944	8.510±2.381	9.935	8.182±2.337	0.853±2.554
11/12	RxnDQN	Greedy	4.978	2.277±1.350	4.978	2.183±1.317	4.978	2.126±1.304	0.038±1.592
11/12	RxnDQN	Random Walk	9.996	8.886±2.449	9.941	8.521±2.390	9.814	8.198±2.340	0.853±2.553
13/14	RxnDQN	Epsilon Greedy	9.989	8.946±2.447	9.917	8.567±2.382	9.898	8.175±2.473	0.888±2.530

13/14	RxnDQN	Greedy	4.978	2.201±1.394	4.978	2.215±1.302	4.978	2.151±1.290	0.032±1.591
13/14	RxnDQN	Random Walk	10.000	8.941±2.457	9.911	8.498±2.387	9.798	8.229±2.338	0.894±2.528
15/16	RxnDQN	Epsilon Greedy	9.994	8.855±2.447	9.991	8.520±2.389	9.983	8.219±2.331	0.896±2.532
15/16	RxnDQN	Greedy	4.978	2.161±1.428	4.978	2.237±1.305	4.978	2.166±1.295	0.031±1.588
15/16	RxnDQN	Random Walk	9.994	8.848±2.440	9.945	8.487±2.375	9.864	8.163±2.316	0.888±2.532
17	RxnMolDQN	Naive	10.000	9.996±0.004	10.000	9.991±0.007	10.000	9.987±0.010	6.216±4.739
18	RxnMolDQN	Bootstrap	10.000	9.996±0.004	10.000	9.992±0.006	9.999	9.989±0.007	6.626±5.037
19	RxnMolDQN	Naive	10.000	9.995±0.005	10.000	9.990±0.007	9.999	9.986±0.009	6.207±4.747
20	RxnMolDQN	Bootstrap	10.000	9.996±0.005	10.000	9.992±0.007	9.999	9.989±0.007	6.660±5.022
21	RxnMolDQN	Naive	10.000	9.994±0.007	10.000	9.990±0.008	9.998	9.986±0.009	6.195±4.726
22	RxnMolDQN	Bootstrap	10.000	9.996±0.003	9.999	9.991±0.007	9.998	9.988±0.008	6.640±5.006
23	RxnMolDQN	Naive	10.000	9.996±0.004	10.000	9.992±0.005	9.999	9.989±0.007	6.190±4.717
24	RxnMolDQN	Bootstrap	10.000	9.996±0.003	9.999	9.992±0.005	9.998	9.989±0.005	6.632±5.005
17/18	RxnMolDQN	Epsilon Greedy	9.992	9.462±0.398	9.850	9.107±0.421	9.811	8.790±0.454	0.667±2.753
17/18	RxnMolDQN	Greedy	9.982	7.744±4.029	9.819	7.291±3.952	9.651	7.163±3.807	0.339±2.763
17/18	RxnMolDQN	Random Walk	10.000	9.439±0.465	9.931	9.056±0.508	9.689	8.799±0.496	0.676±2.746
19/20	RxnMolDQN	Epsilon Greedy	9.999	9.415±0.454	9.961	9.03±0.452	9.815	8.741±0.493	0.657±2.747
19/20	RxnMolDQN	Greedy	9.999	8.439±3.207	9.921	7.999±3.104	9.881	7.694±3.031	0.455±2.646
19/20	RxnMolDQN	Random Walk	9.996	9.419±0.420	9.805	9.046±0.468	9.797	8.79±0.479	0.666±2.741
21/22	RxnMolDQN	Epsilon Greedy	9.999	9.428±1.008	9.905	9.169±0.436	9.627	8.812±0.477	0.709±2.708
21/22	RxnMolDQN	Greedy	9.996	7.870±3.725	9.994	7.546±3.608	9.994	7.246±3.495	0.478±2.648
21/22	RxnMolDQN	Random Walk	9.994	9.508±0.420	9.906	9.103±0.455	9.832	8.868±0.489	0.711±2.716
23/24	RxnMolDQN	Epsilon Greedy	9.992	9.527±0.355	9.827	8.976±1.020	9.764	8.781±0.480	0.709±2.719
23/24	RxnMolDQN	Greedy	9.999	8.06±3.643	9.967	7.632±3.610	9.842	7.361±3.507	0.575±2.632
23/24	RxnMolDQN	Random Walk	9.998	9.356±0.494	9.909	9.000±0.523	9.720	8.715±0.508	0.713±2.715

(a)



(b)



(c)

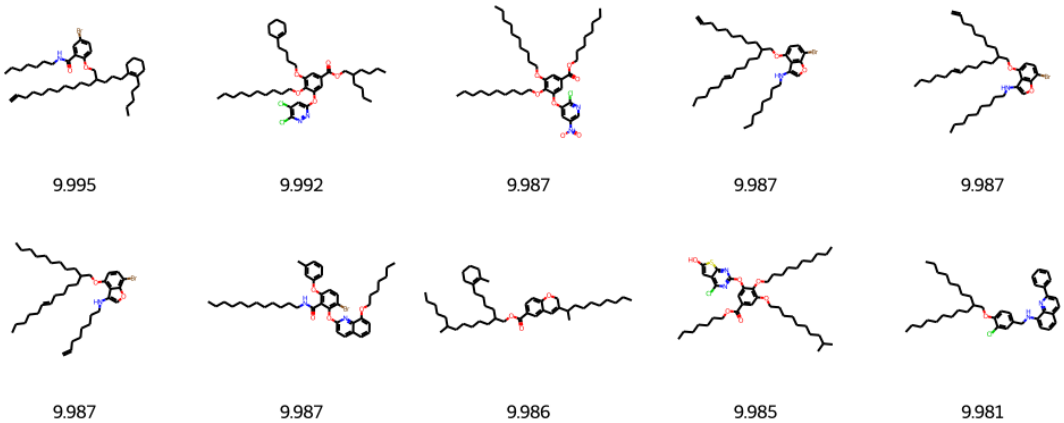


Figure F13: Top 10 penalized logP scored compounds when using flucloxacillin/DB00301 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

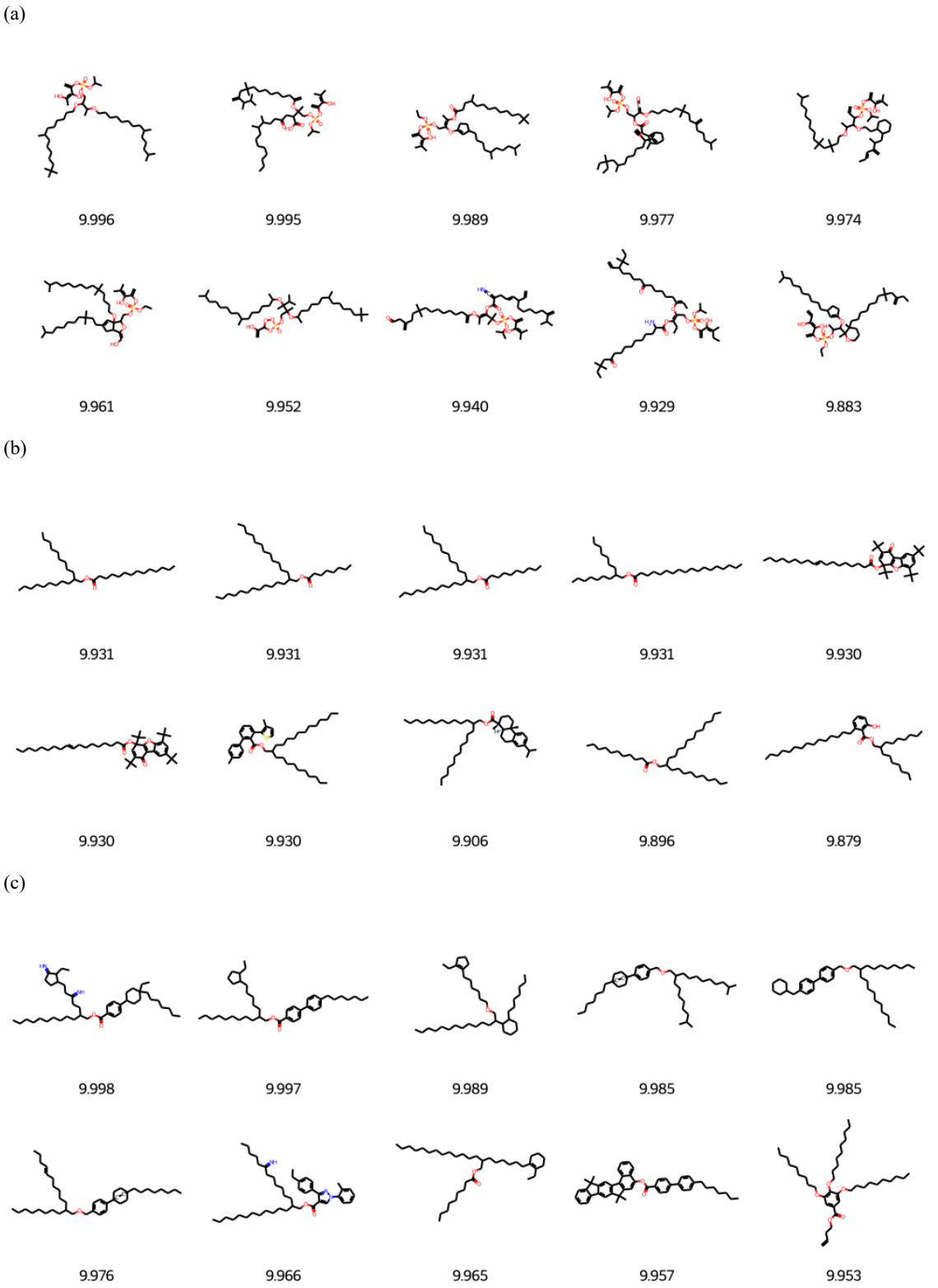
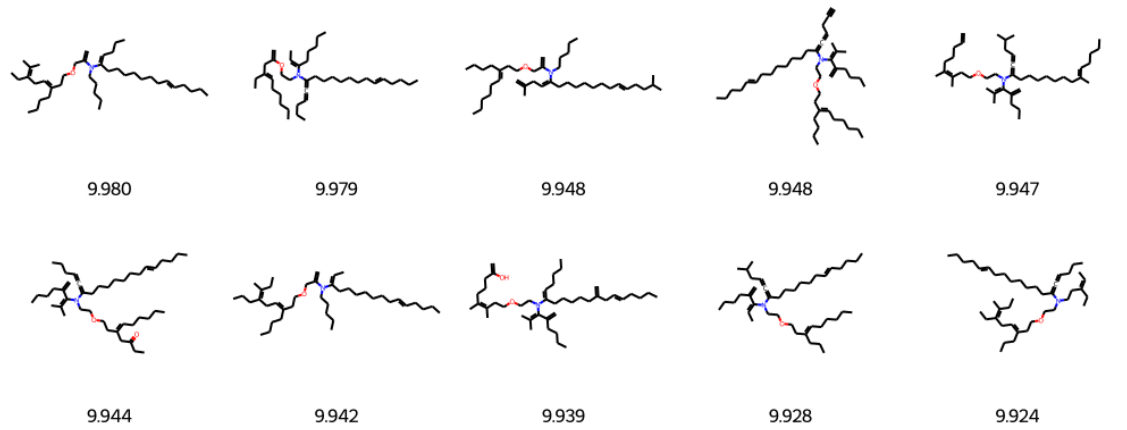
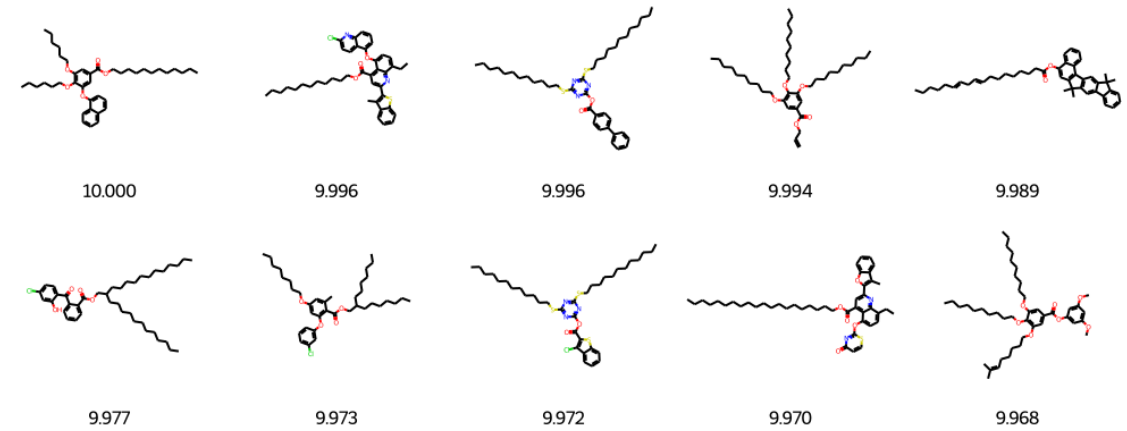


Figure F14: Top 10 penalized logP scored compounds when using etravirine/DB06414 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

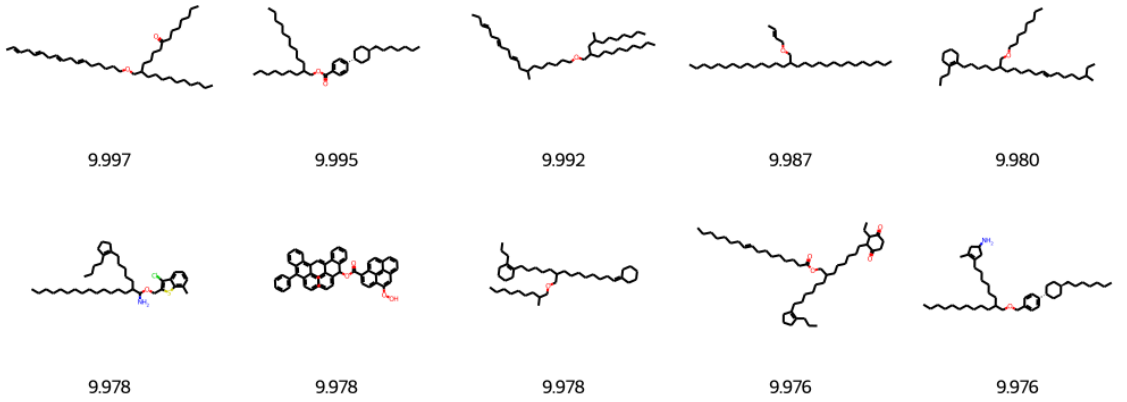
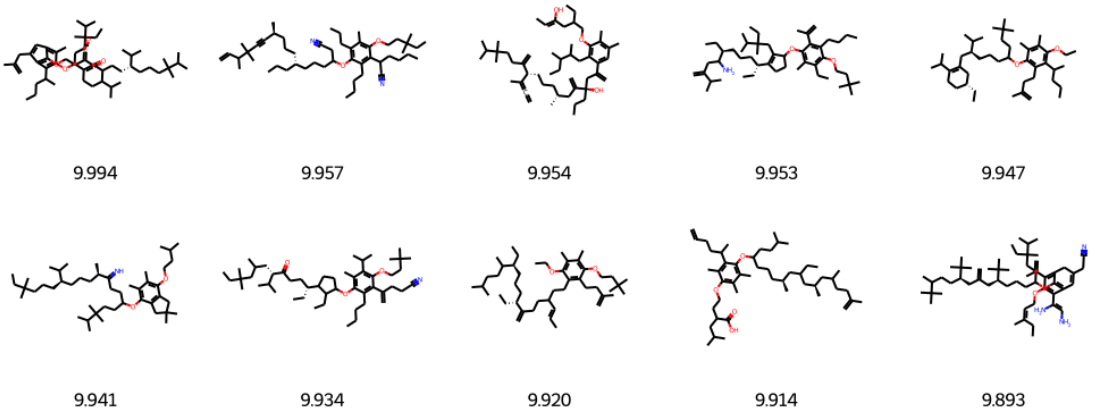
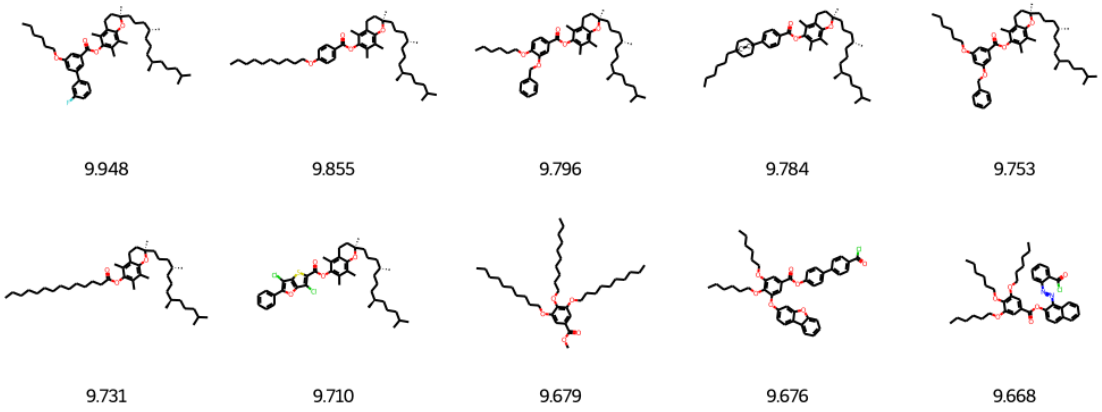


Figure F15: Top 10 penalized logP scored compounds when using sofosbuvir/DB08934 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

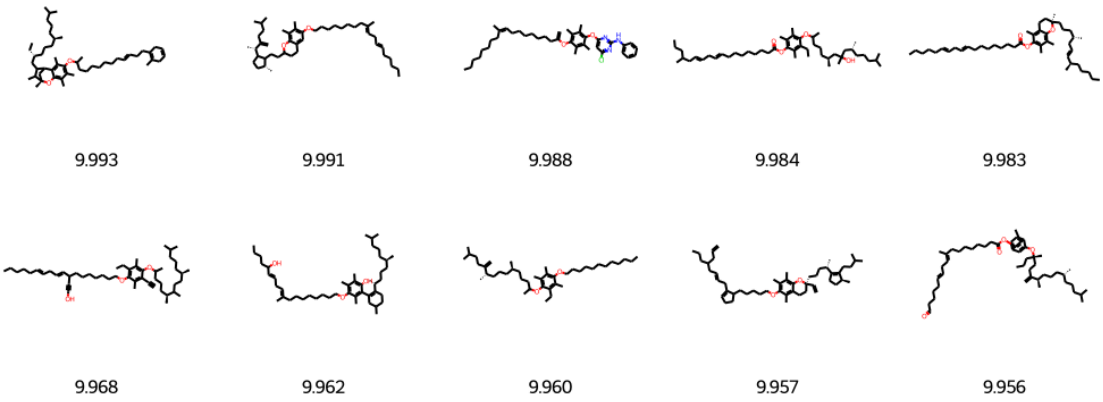
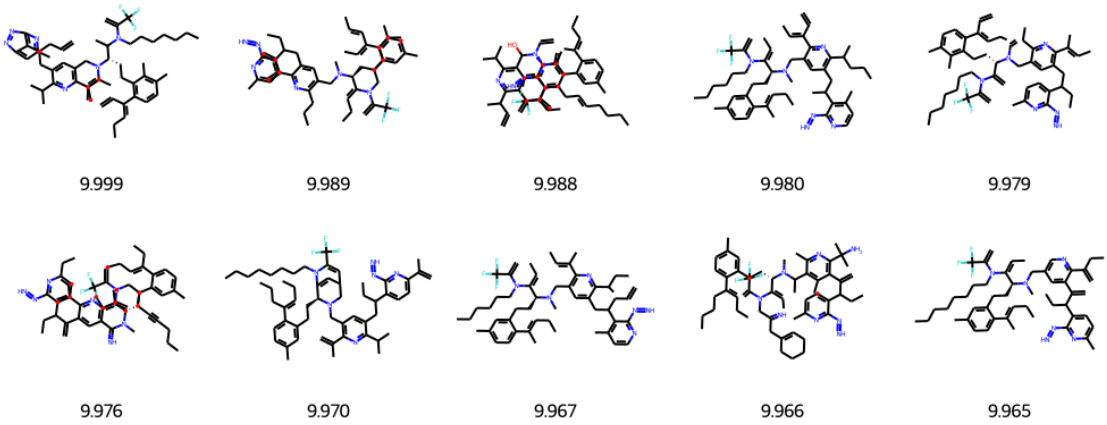
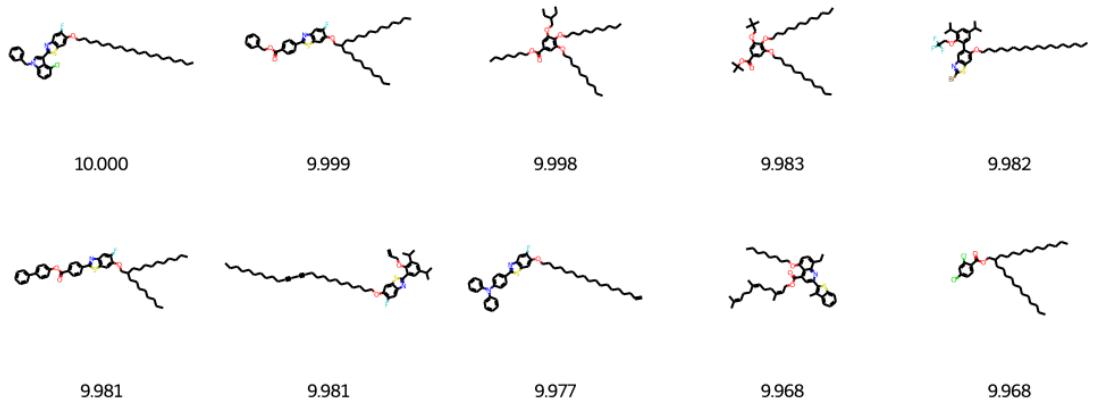


Figure F16: Top 10 penalized logP scored compounds when using cisapride/DB00604 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

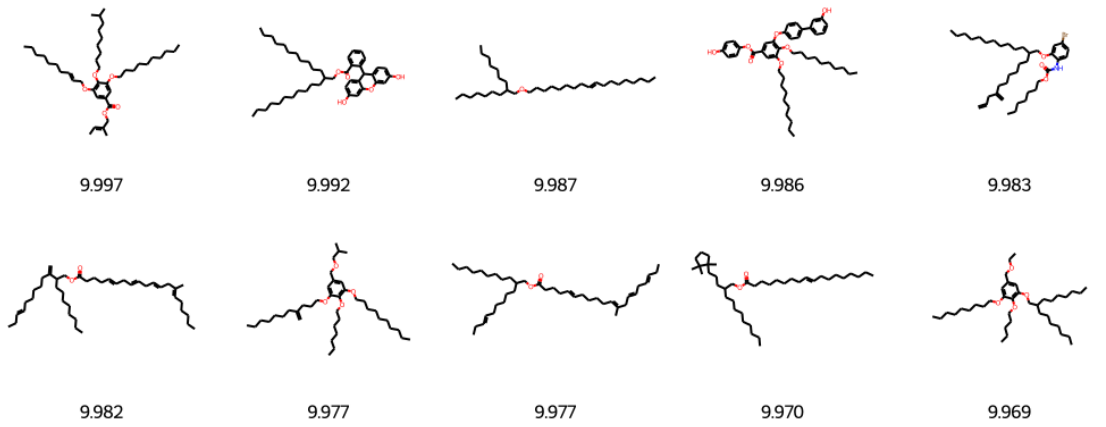


Figure F17: Top 10 penalized logP scored compounds when using zopiclone/DB00198 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

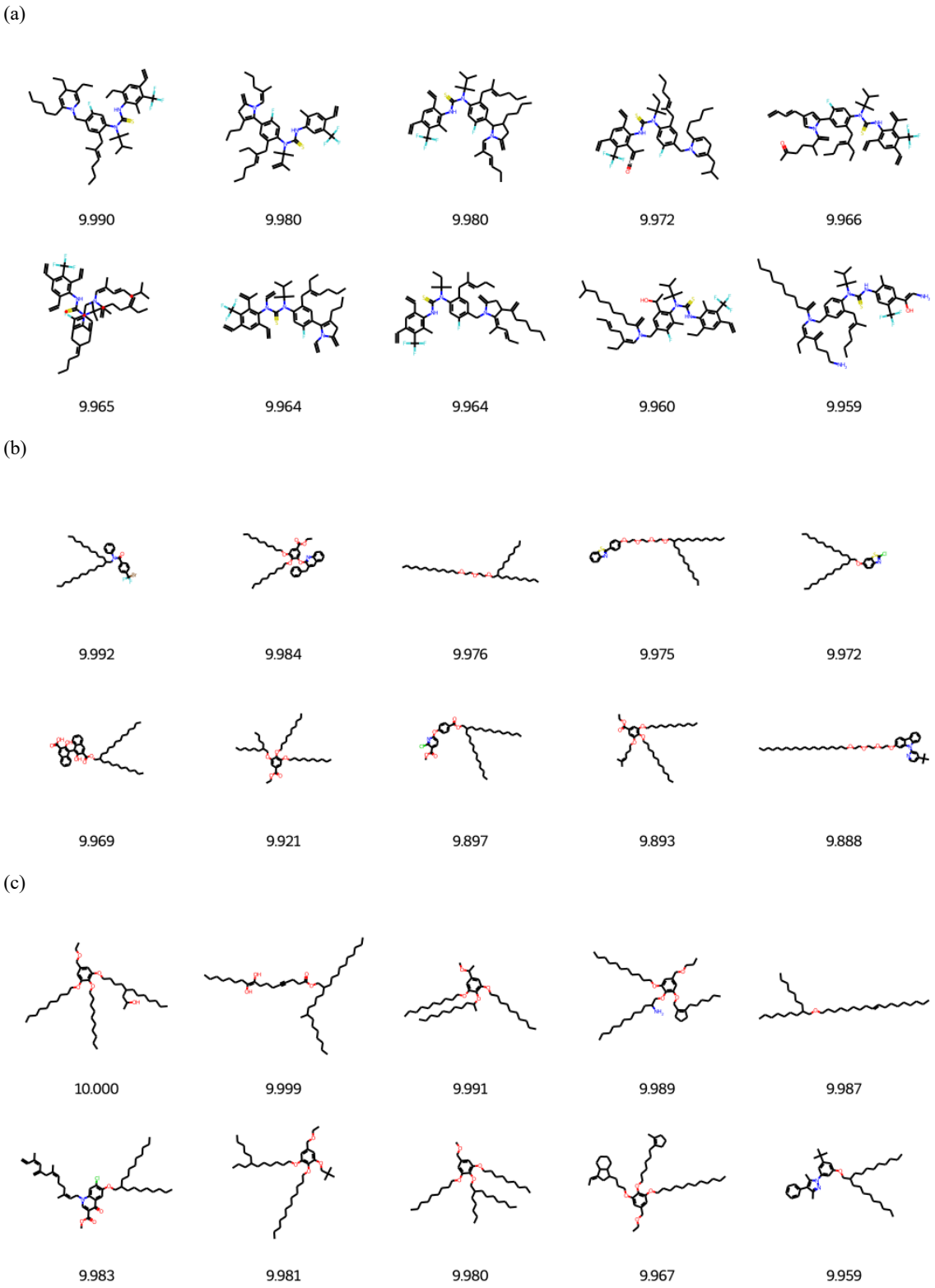
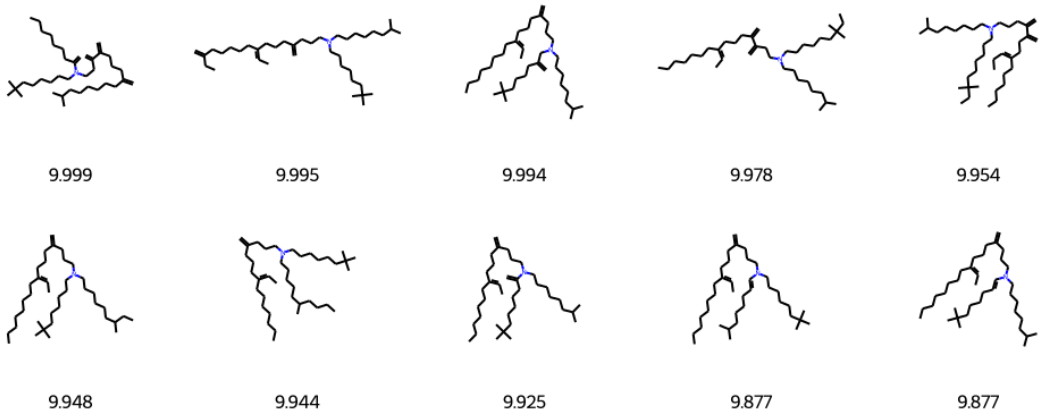
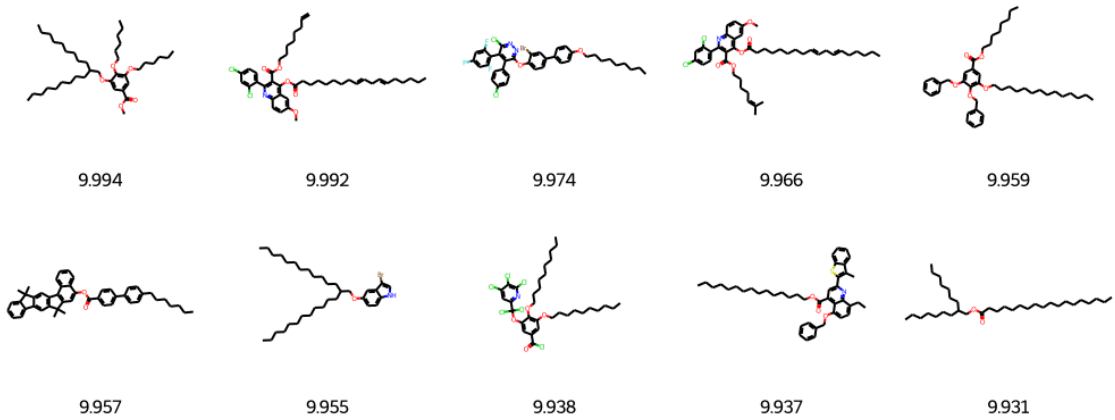


Figure F18: Top 10 penalized logP scored compounds when using phthalylsulfathiazole/DB13248 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

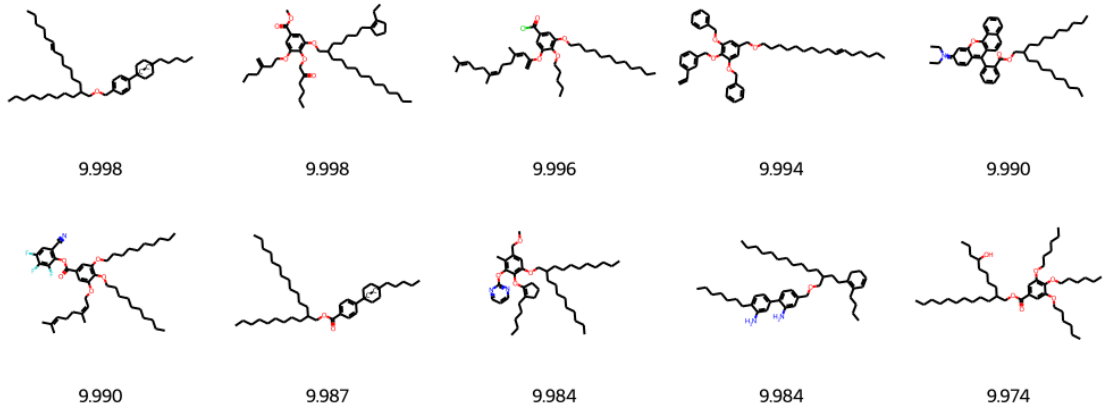


Figure F19: Top 10 penalized logP scored compounds when using ibutilide/DB00308 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

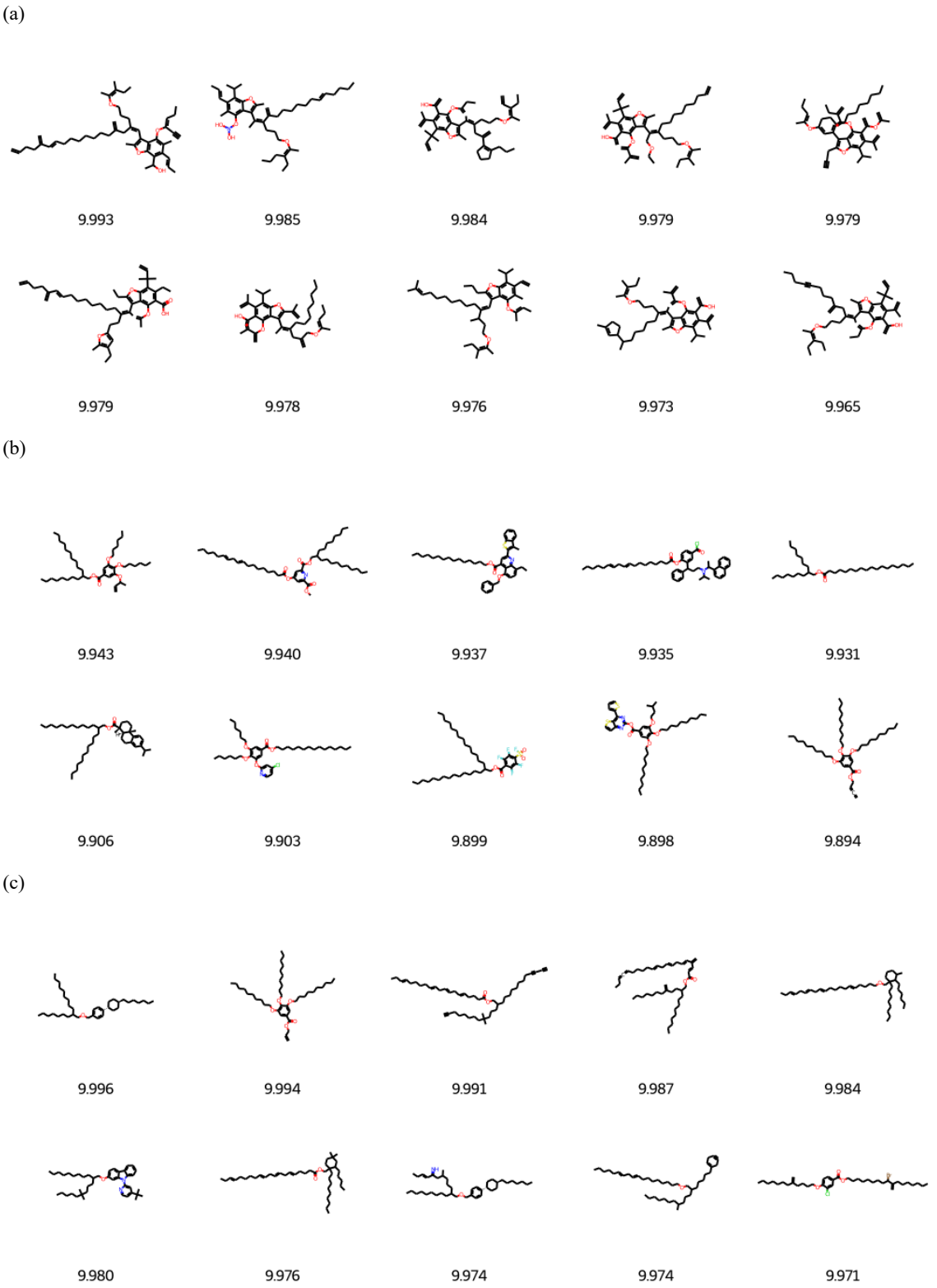
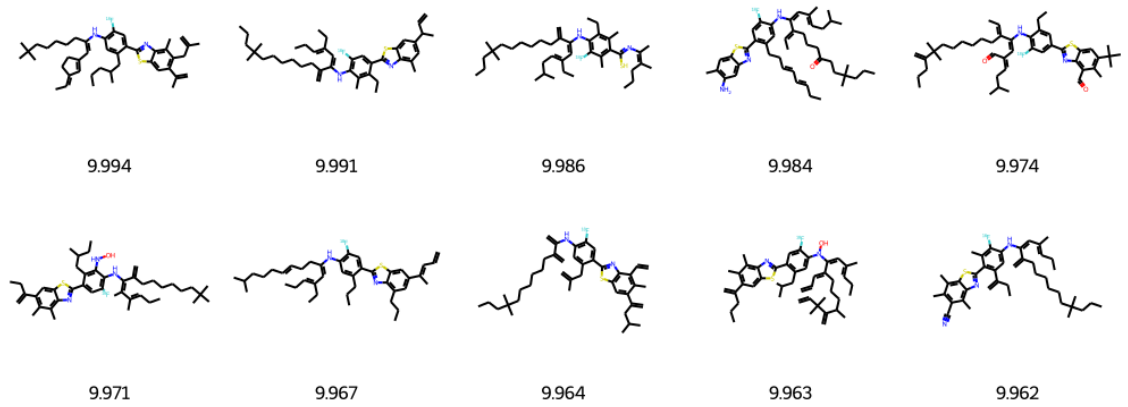
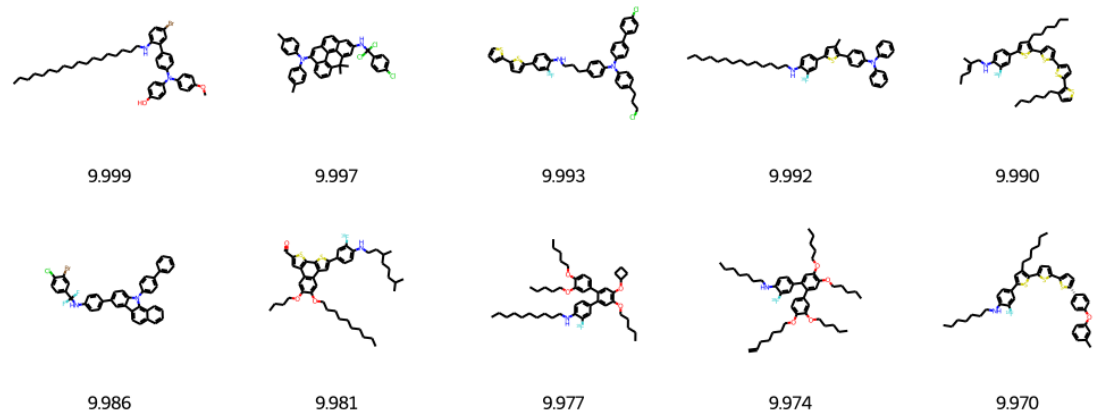


Figure F20: Top 10 penalized logP scored compounds when using acetohexamide/DB00414 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

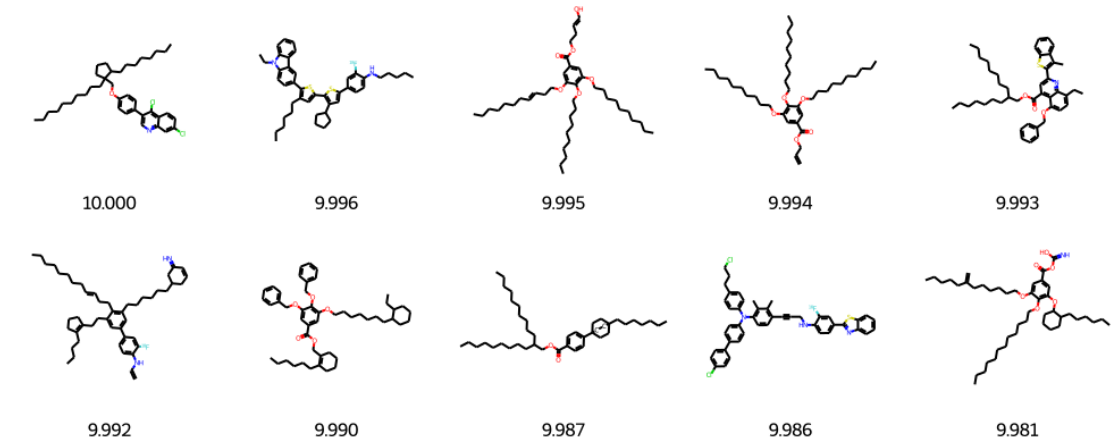
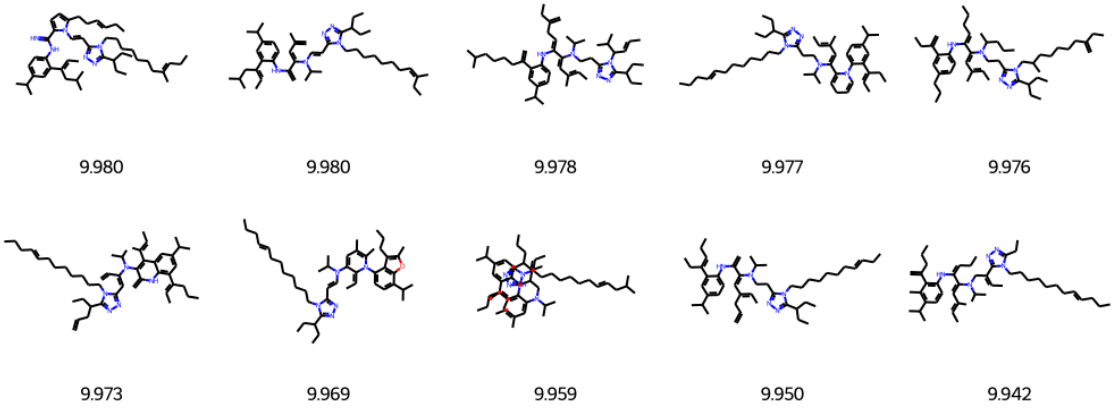
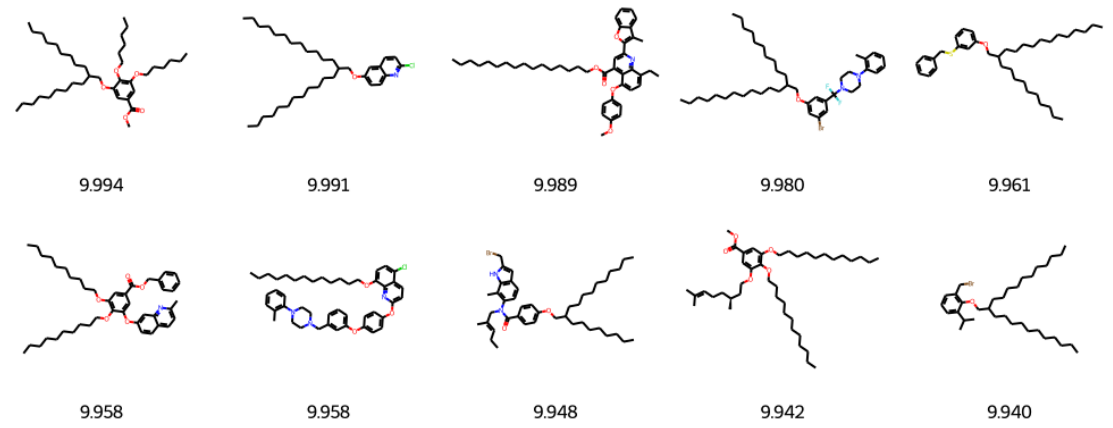


Figure F21: Top 10 penalized logP scored compounds when using creatine/DB00148 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

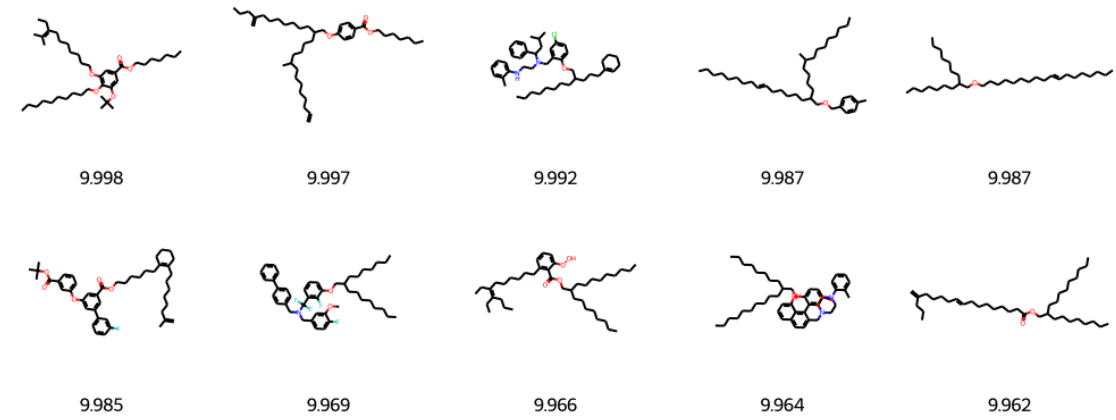


Figure F22: Top 10 penalized logP scored compounds when using ponatinib/DB08901 from DrugBank (Wishart *et al.*, 2018) as a starting molecule where the reward function is penalized logP: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

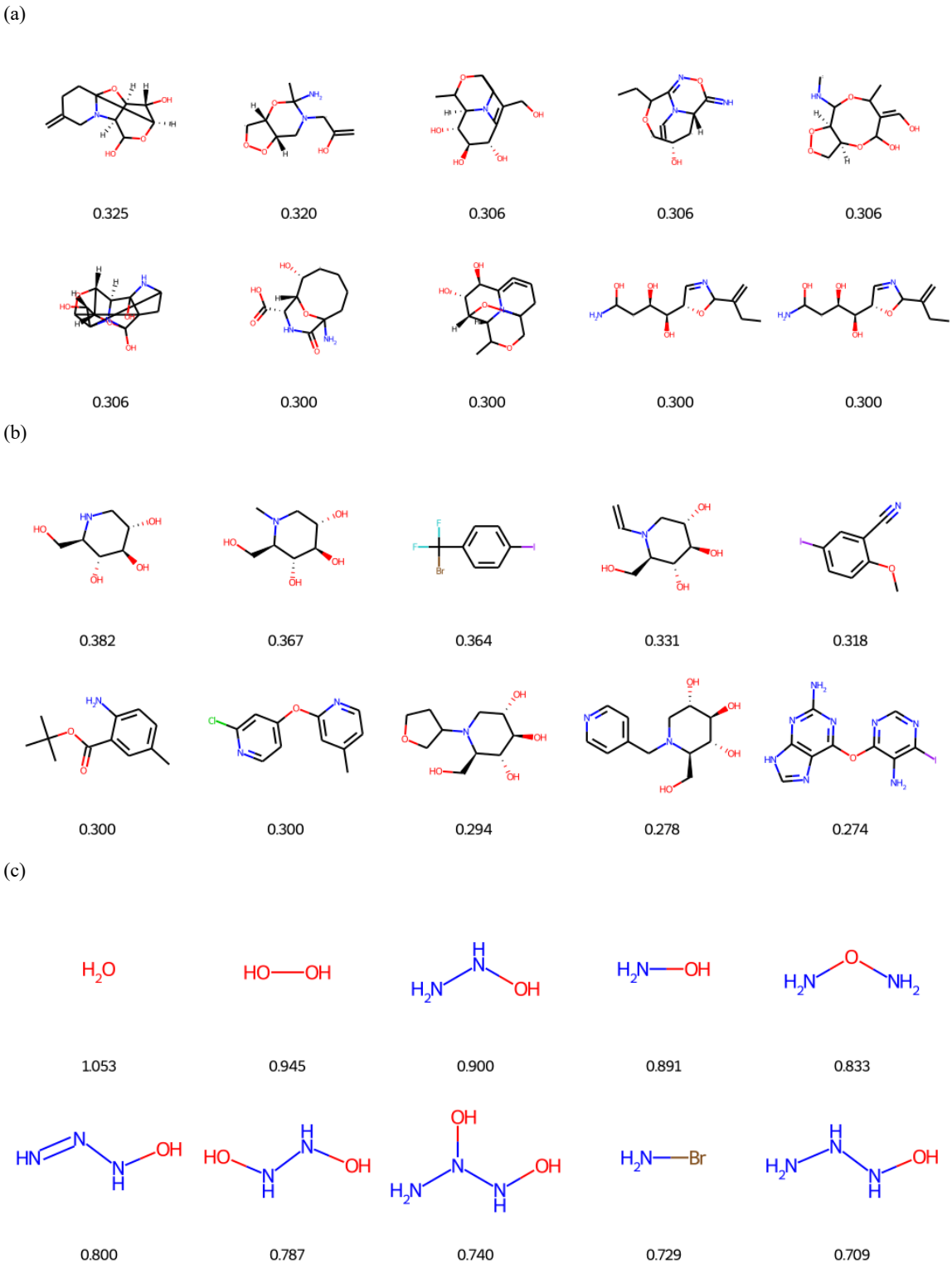
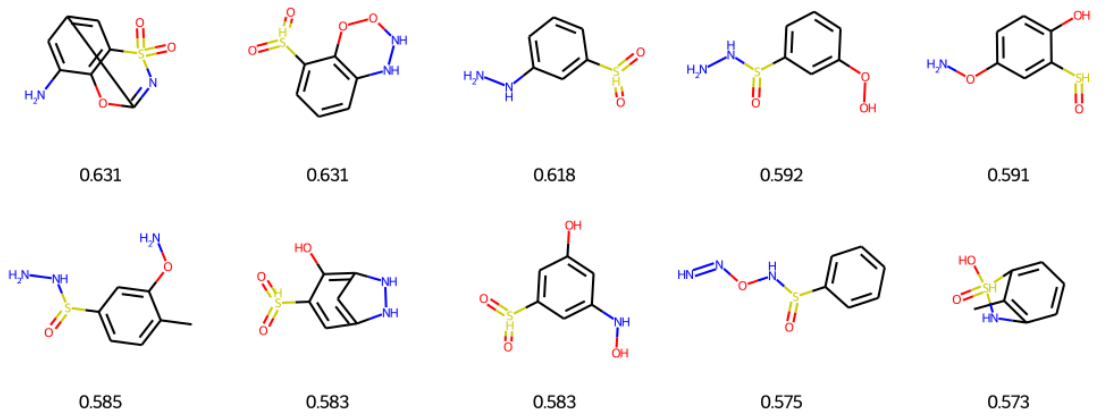
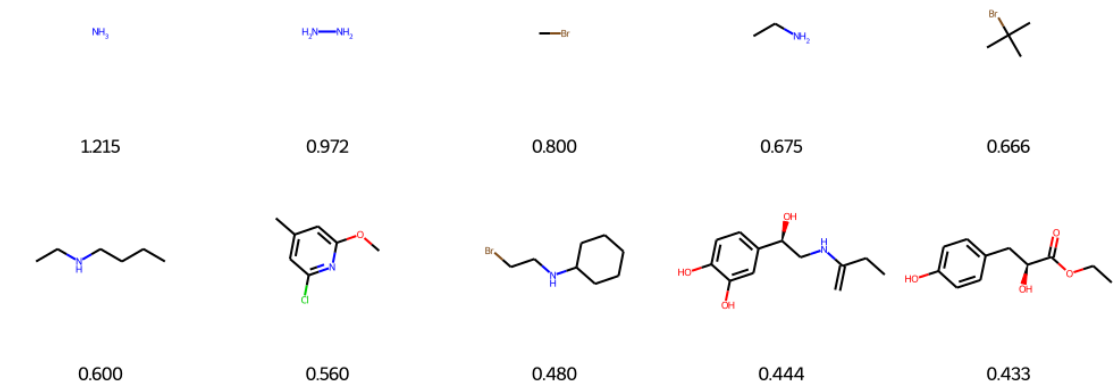


Figure F23: Top 10 predicted ligand efficiency scored compounds when using NBV/2v3d from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is predicted ligand efficiency: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

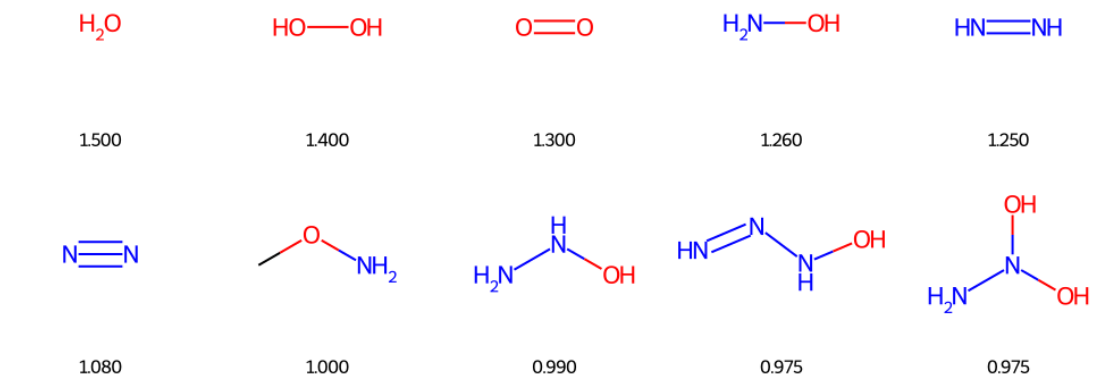
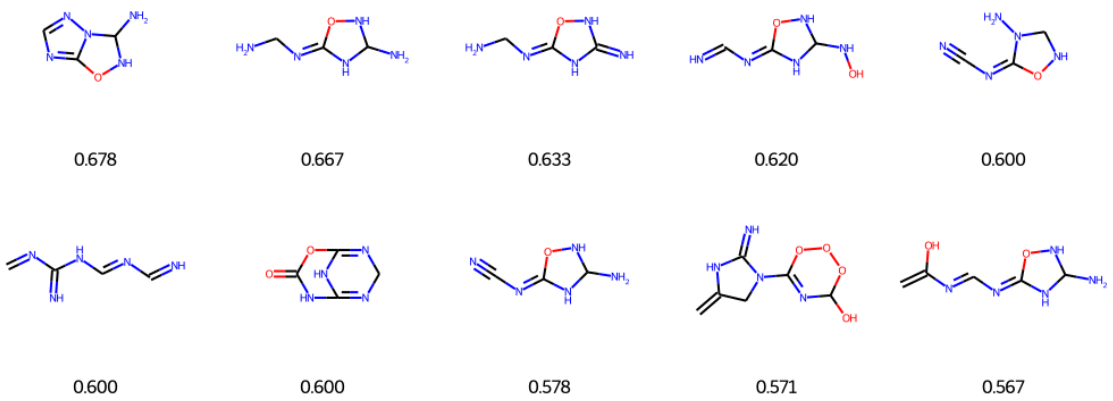
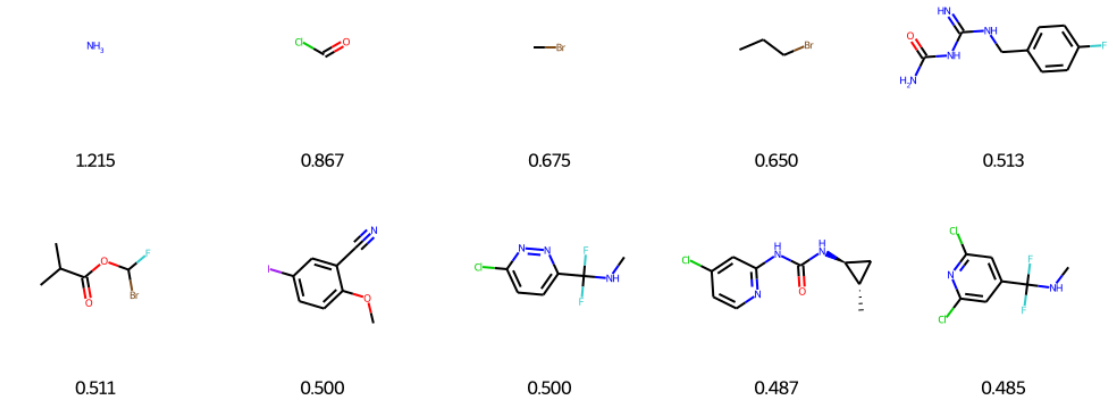


Figure F24: Top 10 predicted ligand efficiency scored compounds when using FB2/2wej from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is predicted ligand efficiency: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

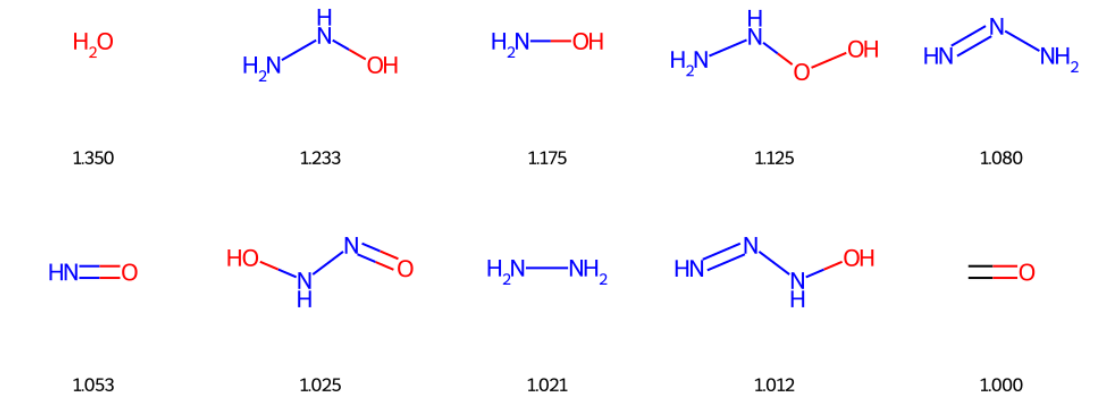


Figure F25: Top 10 predicted ligand efficiency scored compounds when using XRG/3ch9 from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is predicted ligand efficiency: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

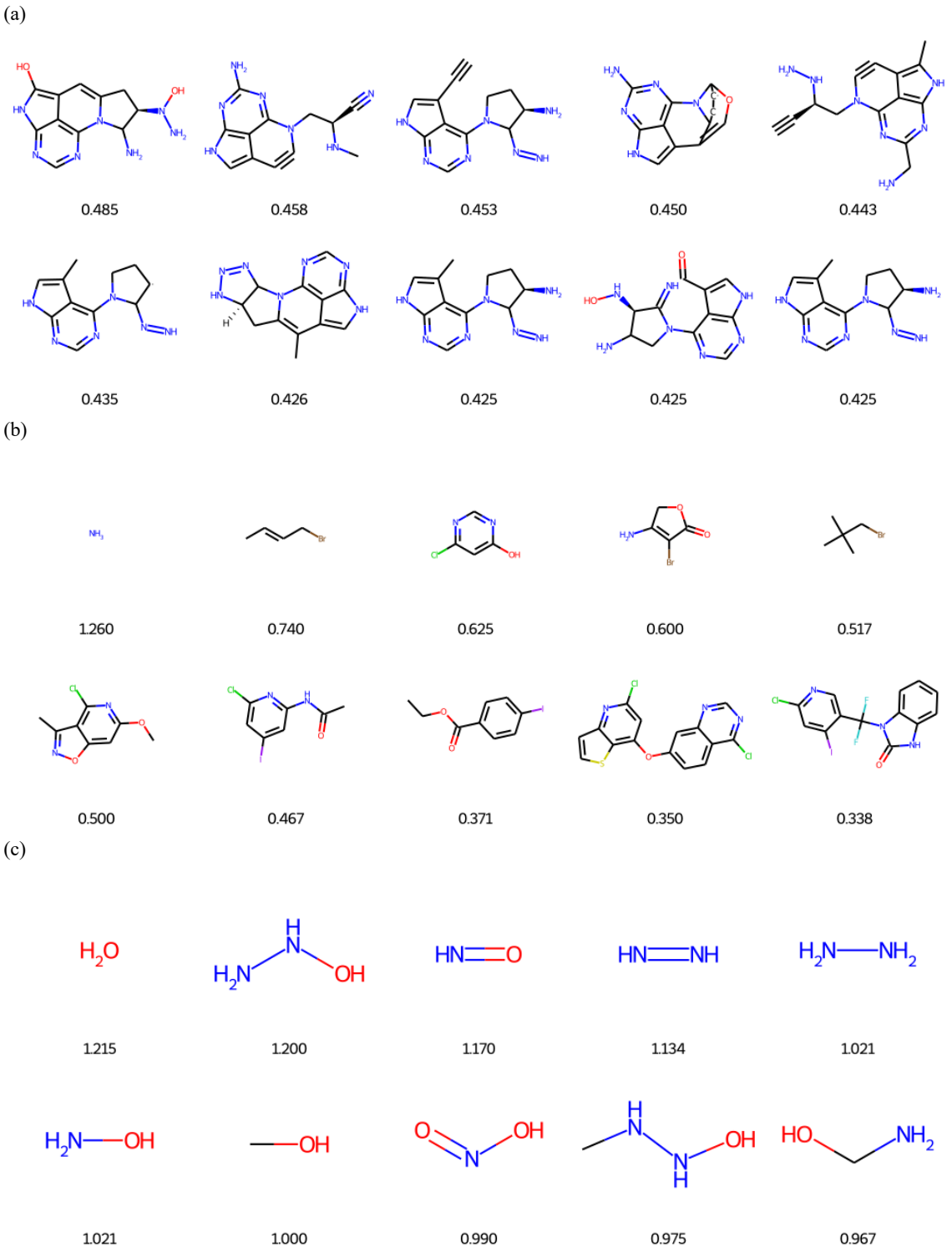
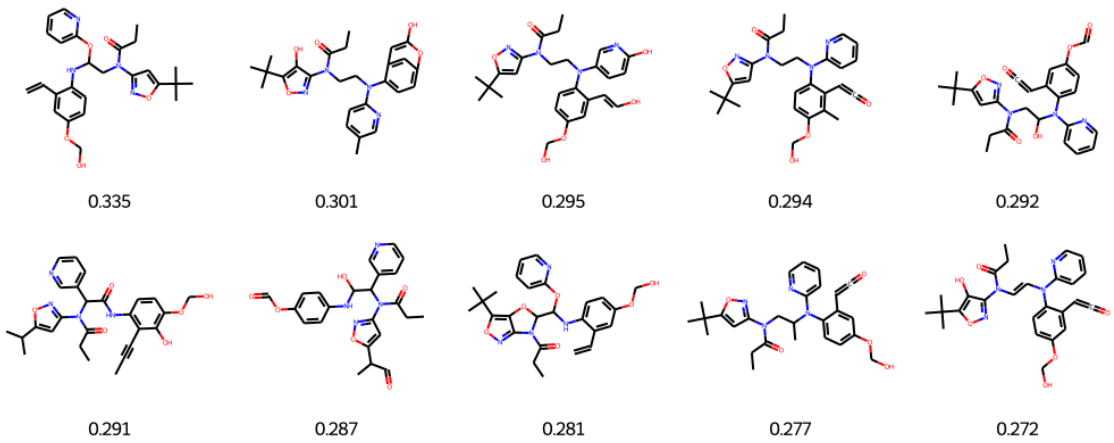
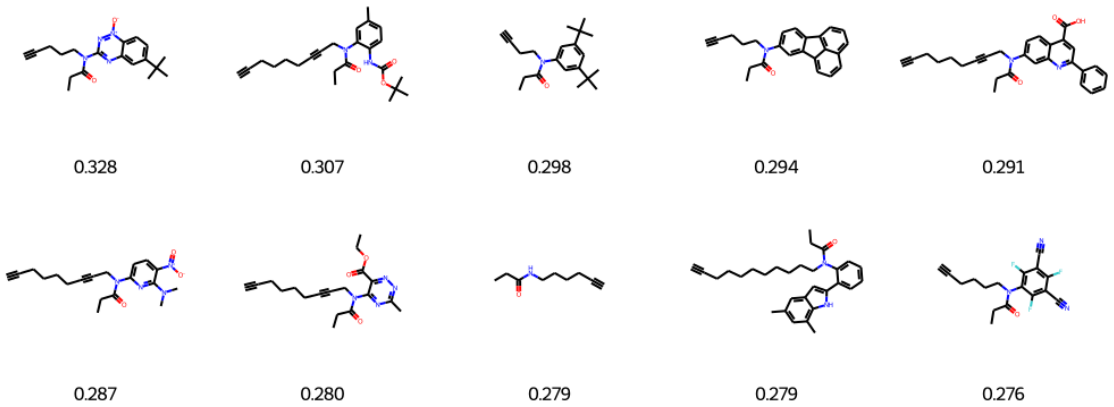


Figure F26: Top 10 predicted ligand efficiency scored compounds when using XFE/3mv5 from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is predicted ligand efficiency: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

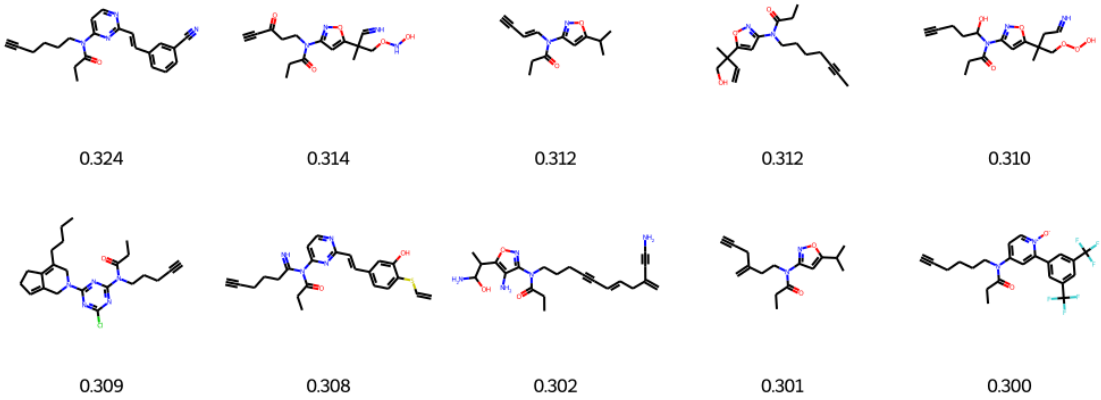


Figure F27: Top 10 predicted ligand efficiency scored compounds when using Mpro-x2694_0A from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) as a starting molecule where the reward function is predicted ligand efficiency: (a) c3D-MolDQN, (b) c3D-RxnDQN, and (c) c3D-RxnMolDQN.

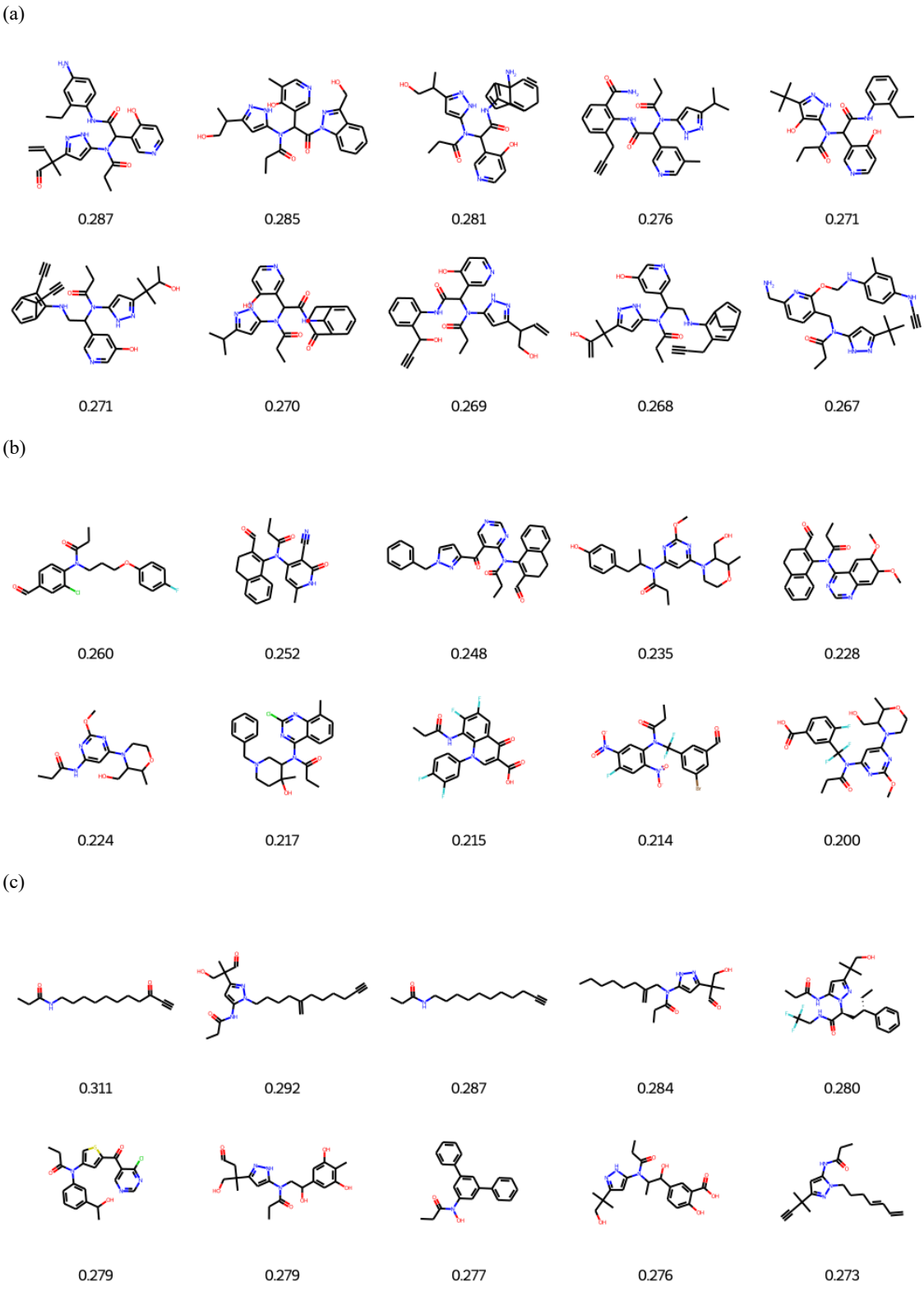
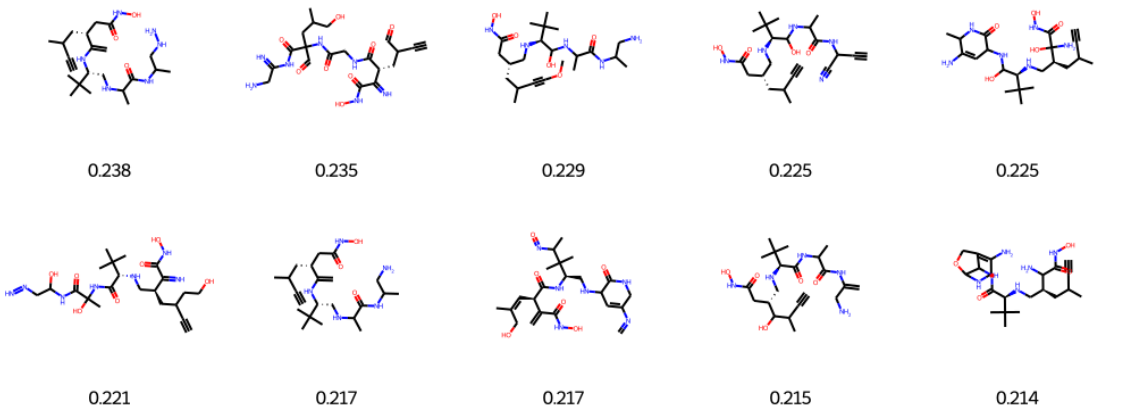
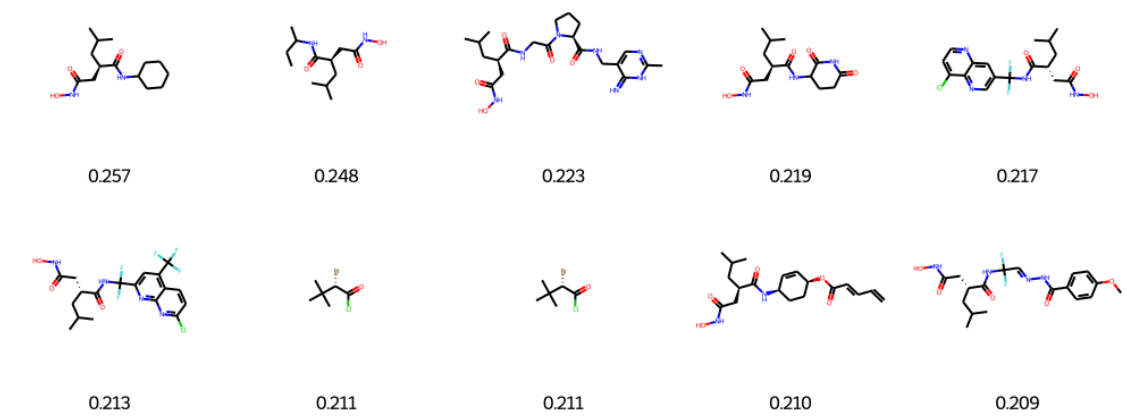


Figure F28: Top 10 predicted ligand efficiency scored compounds when using Mpro-x2705_0A from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) as a starting molecule where the reward function is predicted ligand efficiency: (a) c3D-MolDQN, (b) c3D-RxnDQN, and (c) c3D-RxnMolDQN.

(a)



(b)



(c)

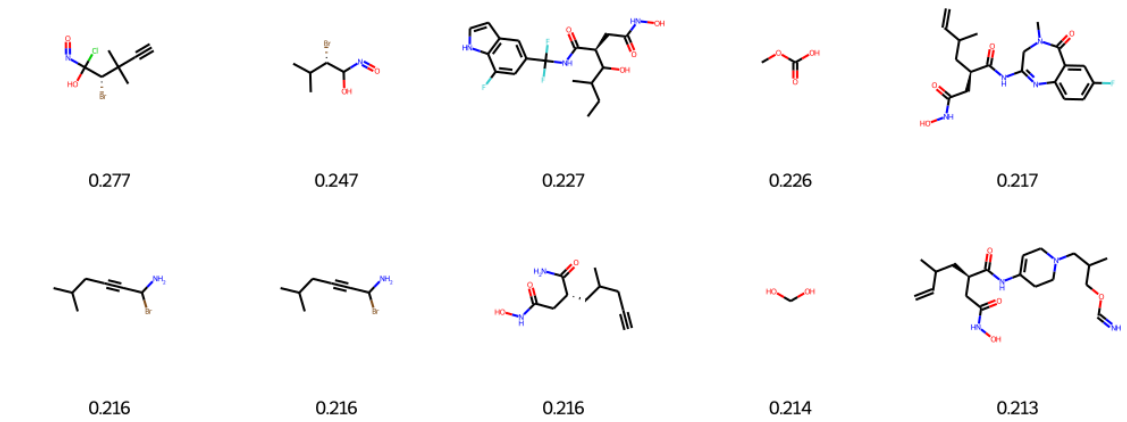
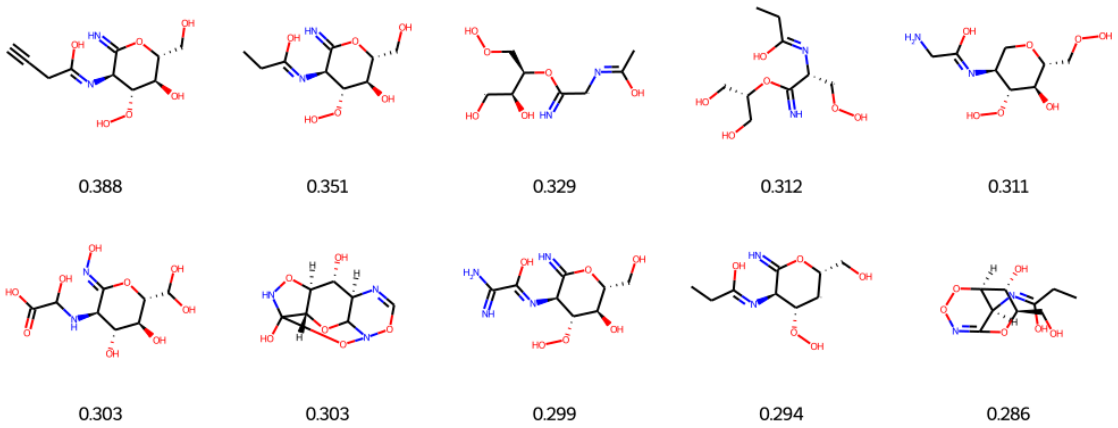
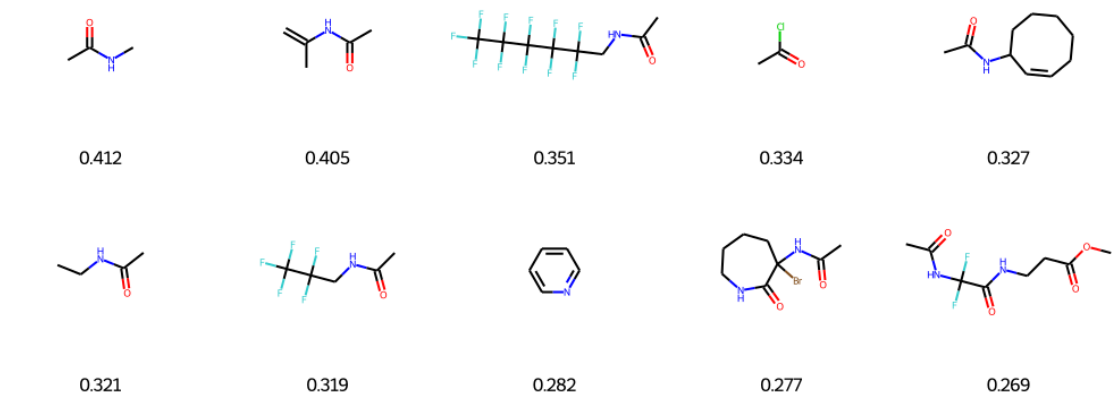


Figure F29: Top 10 normalized SuCOS scored compounds when using INN/3kme from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

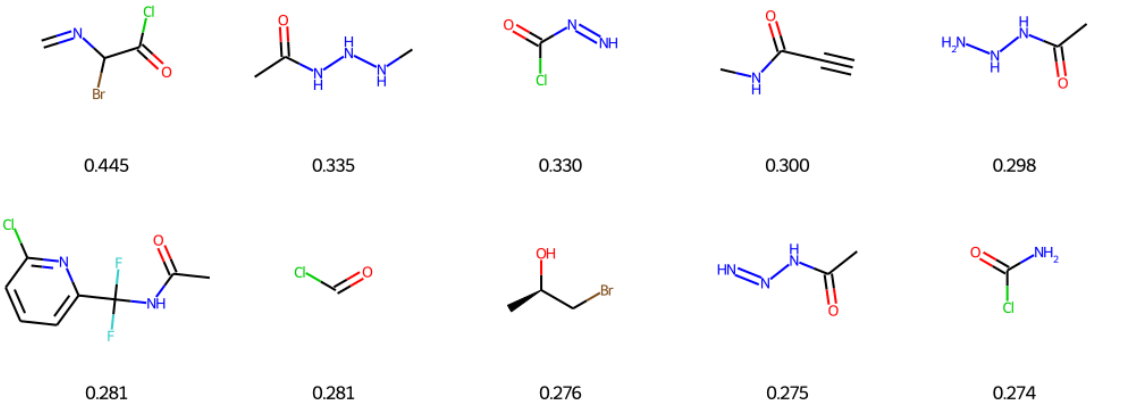
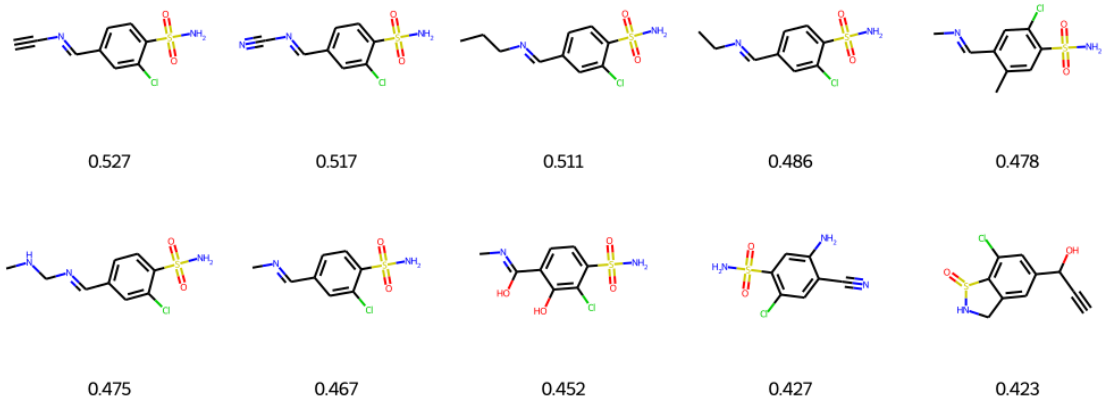
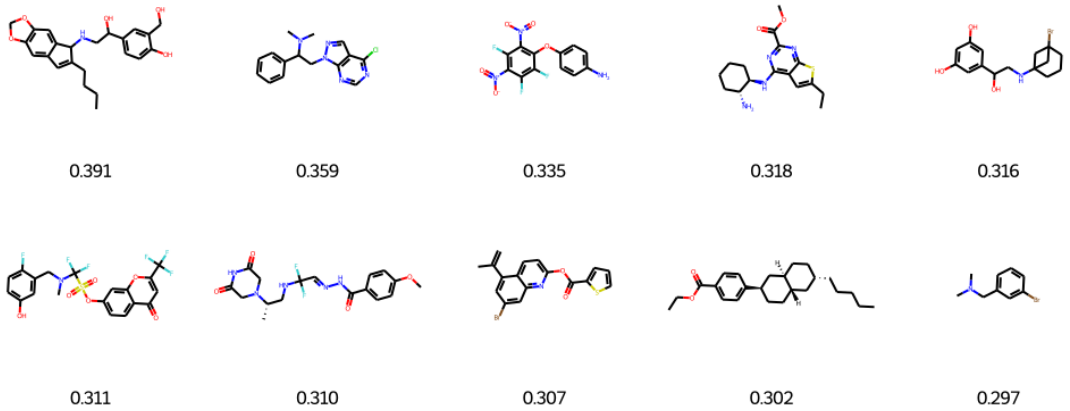


Figure F30: Top 10 normalized SuCOS scored compounds when using LOG/2xm2 from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

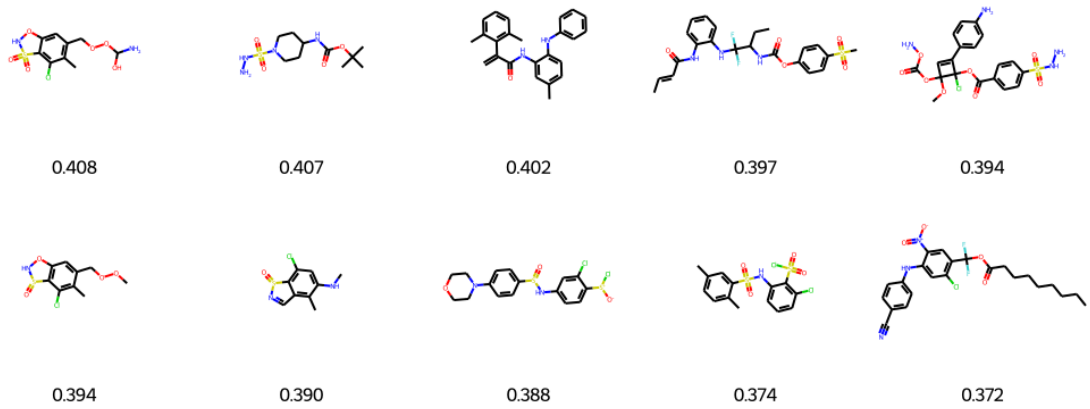
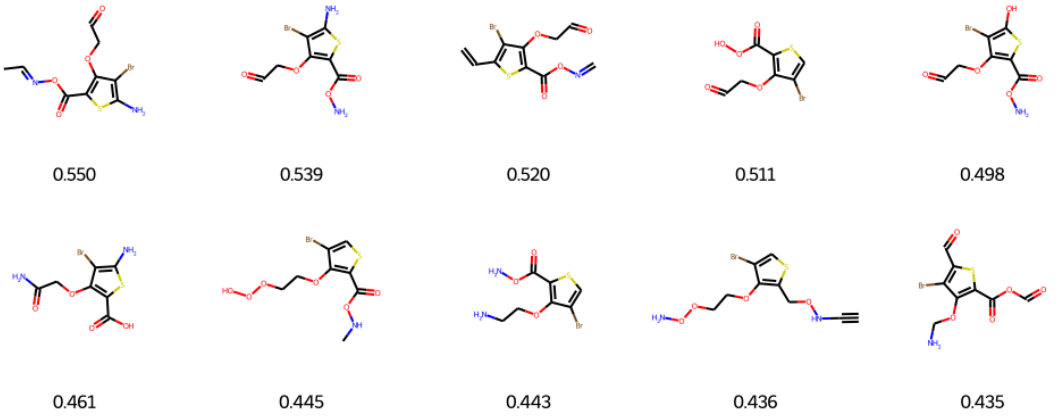
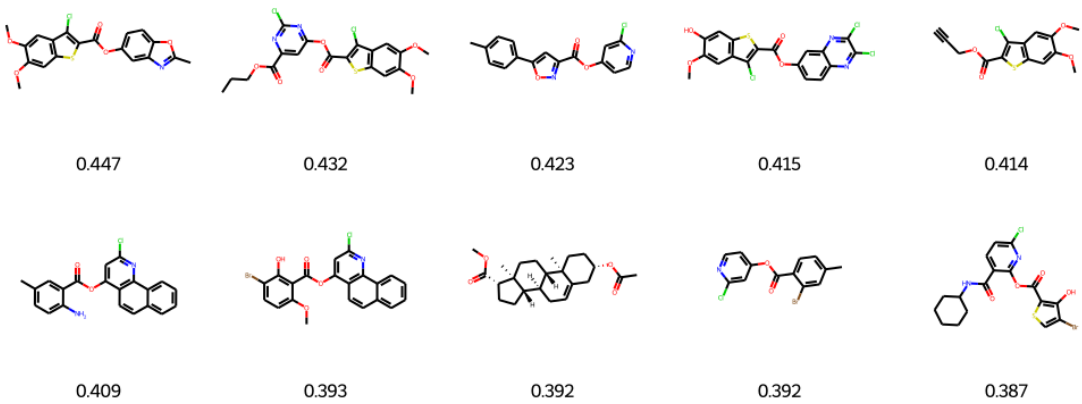


Figure F31: Top 10 normalized SuCOS scored compounds when using FB1/2weh from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

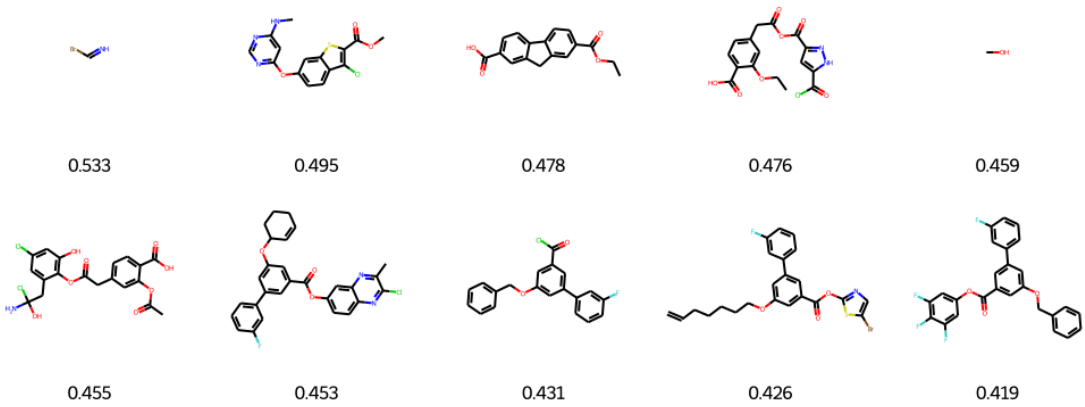
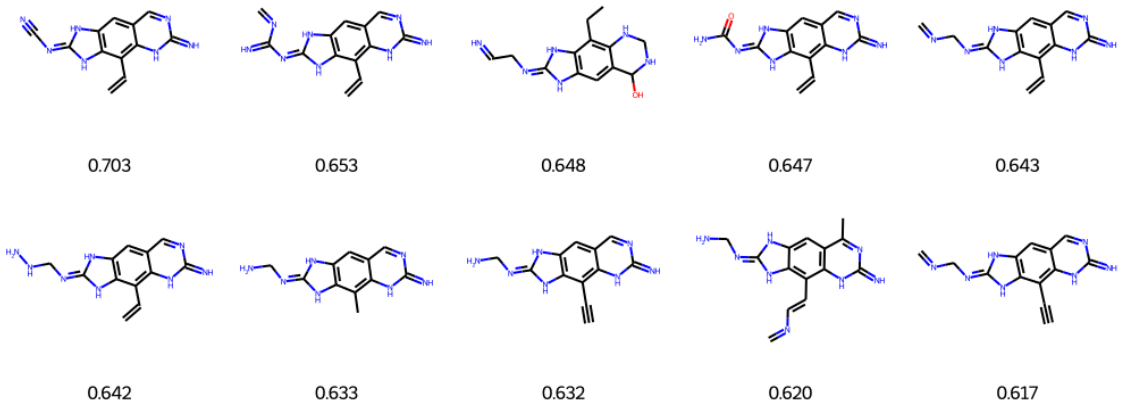
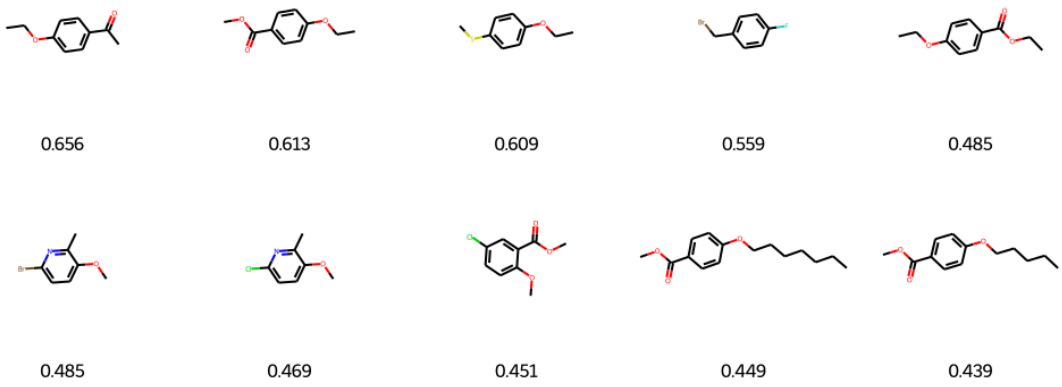


Figure F32: Top 10 normalized SuCOS scored compounds when using 512/2hb1 from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

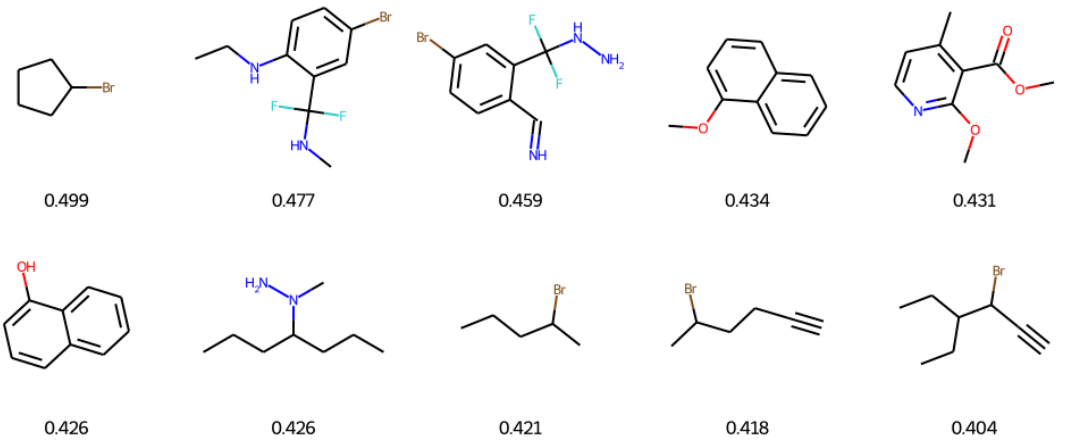


Figure F33: Top 10 normalized SuCOS scored compounds when using 2MQ/3gc5 from the Malhotra set (Malhotra and Karanicolas, 2017) as a starting molecule where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

Appendix G: Walters' Compound Quality

Table G1: Walters' compound quality for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, QED. The values are approximated to three decimal places. Average Walters' compound quality is computed by taking an average of Walters' compound quality for all molecules generated across 100 DrugBank compounds.

Method Number	Method	Baseline/DQN Methods	Average Walters' Compound Quality (/1)
1	MolDQN	Naive	0.175
2	MolDQN	Bootstrap	0.214
3	MolDQN	Naive	0.163
4	MolDQN	Bootstrap	0.164
5	MolDQN	Naive	0.183±0.097
6	MolDQN	Bootstrap	0.218±0.082
7	MolDQN	Naive	0.174±0.100
8	MolDQN	Bootstrap	0.213±0.079
1/2	MolDQN	Epsilon Greedy	0.019
1/2	MolDQN	Greedy	1.000
1/2	MolDQN	Random Walk	0.021
3/4	MolDQN	Epsilon Greedy	0.018
3/4	MolDQN	Greedy	1.000
3/4	MolDQN	Random Walk	0.021
5/6	MolDQN	Epsilon Greedy	0.003±0.003
5/6	MolDQN	Greedy	0.570±0.495
5/6	MolDQN	Random Walk	0.003±0.003
7/8	MolDQN	Epsilon Greedy	0.003±0.003
7/8	MolDQN	Greedy	0.580±0.494
7/8	MolDQN	Random Walk	0.003±0.002
9	RxnDQN	Naive	0.635±0.178
10	RxnDQN	Bootstrap	0.608±0.177
11	RxnDQN	Naive	0.633±0.176
12	RxnDQN	Bootstrap	0.614±0.171
13	RxnDQN	Naive	0.632±0.176
14	RxnDQN	Bootstrap	0.609±0.17
15	RxnDQN	Naive	0.624±0.178
16	RxnDQN	Bootstrap	0.610±0.174
9/10	RxnDQN	Epsilon Greedy	0.401±0.160
9/10	RxnDQN	Greedy	0.725±0.304
9/10	RxnDQN	Random Walk	0.400±0.159
11/12	RxnDQN	Epsilon Greedy	0.399±0.160
11/12	RxnDQN	Greedy	0.725±0.304
11/12	RxnDQN	Random Walk	0.401±0.159
13/14	RxnDQN	Epsilon Greedy	0.401±0.16
13/14	RxnDQN	Greedy	0.726±0.303
13/14	RxnDQN	Random Walk	0.401±0.159
15/16	RxnDQN	Epsilon Greedy	0.400±0.159
15/16	RxnDQN	Greedy	0.726±0.302

15/16	RxnDQN	Random Walk	0.399±0.160
17	RxnMolDQN	Naive	0.602±0.123
18	RxnMolDQN	Bootstrap	0.585±0.063
19	RxnMolDQN	Naive	0.600±0.110
20	RxnMolDQN	Bootstrap	0.582±0.069
21	RxnMolDQN	Naive	0.594±0.126
22	RxnMolDQN	Bootstrap	0.588±0.061
23	RxnMolDQN	Naive	0.610±0.115
24	RxnMolDQN	Bootstrap	0.585±0.057
17/18	RxnMolDQN	Epsilon Greedy	0.295±0.047
17/18	RxnMolDQN	Greedy	0.315±0.187
17/18	RxnMolDQN	Random Walk	0.293±0.047
19/20	RxnMolDQN	Epsilon Greedy	0.293±0.046
19/20	RxnMolDQN	Greedy	0.335±0.196
19/20	RxnMolDQN	Random Walk	0.295±0.046
21/22	RxnMolDQN	Epsilon Greedy	0.294±0.046
21/22	RxnMolDQN	Greedy	0.317±0.158
21/22	RxnMolDQN	Random Walk	0.294±0.047
23/24	RxnMolDQN	Epsilon Greedy	0.292±0.047
23/24	RxnMolDQN	Greedy	0.307±0.166
23/24	RxnMolDQN	Random Walk	0.293±0.047

Table G2: Walters' compound quality for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, penalized logP. The values are approximated to three decimal places.

Method Number	Method	Baseline/DQN Methods	Average Walters' Compound Quality (/1)
1	MolDQN	Naive	0.217
2	MolDQN	Bootstrap	0.350
3	MolDQN	Naive	0.117
4	MolDQN	Bootstrap	0.271
5	MolDQN	Naive	0.025±0.045
6	MolDQN	Bootstrap	0.030±0.048
7	MolDQN	Naive	0.026±0.048
8	MolDQN	Bootstrap	0.028±0.046
1/2	MolDQN	Epsilon Greedy	0.017
1/2	MolDQN	Greedy	1.000
1/2	MolDQN	Random Walk	0.020
3/4	MolDQN	Epsilon Greedy	0.017
3/4	MolDQN	Greedy	1.000
3/4	MolDQN	Random Walk	0.016
5/6	MolDQN	Epsilon Greedy	0.003±0.003
5/6	MolDQN	Greedy	0.550±0.497
5/6	MolDQN	Random Walk	0.003±0.003
7/8	MolDQN	Epsilon Greedy	0.003±0.003
7/8	MolDQN	Greedy	0.580±0.494
7/8	MolDQN	Random Walk	0.003±0.003
9	RxnDQN	Naive	0.285±0.166
10	RxnDQN	Bootstrap	0.296±0.167
11	RxnDQN	Naive	0.285±0.165
12	RxnDQN	Bootstrap	0.295±0.163
13	RxnDQN	Naive	0.284±0.168
14	RxnDQN	Bootstrap	0.293±0.163
15	RxnDQN	Naive	0.288±0.165
16	RxnDQN	Bootstrap	0.290±0.164
9/10	RxnDQN	Epsilon Greedy	0.400±0.159
9/10	RxnDQN	Greedy	0.723±0.305
9/10	RxnDQN	Random Walk	0.400±0.160
11/12	RxnDQN	Epsilon Greedy	0.400±0.158
11/12	RxnDQN	Greedy	0.725±0.305
11/12	RxnDQN	Random Walk	0.400±0.159
13/14	RxnDQN	Epsilon Greedy	0.401±0.159
13/14	RxnDQN	Greedy	0.726±0.304
13/14	RxnDQN	Random Walk	0.401±0.158
15/16	RxnDQN	Epsilon Greedy	0.399±0.160
15/16	RxnDQN	Greedy	0.725±0.303
15/16	RxnDQN	Random Walk	0.400±0.160
17	RxnMolDQN	Naive	0.239±0.097
18	RxnMolDQN	Bootstrap	0.242±0.063
19	RxnMolDQN	Naive	0.236±0.092
20	RxnMolDQN	Bootstrap	0.243±0.063
21	RxnMolDQN	Naive	0.232±0.094

22	RxnMolDQN	Bootstrap	0.247±0.062
23	RxnMolDQN	Naive	0.236±0.094
24	RxnMolDQN	Bootstrap	0.245±0.062
17/18	RxnMolDQN	Epsilon Greedy	0.294±0.045
17/18	RxnMolDQN	Greedy	0.339±0.233
17/18	RxnMolDQN	Random Walk	0.295±0.046
19/20	RxnMolDQN	Epsilon Greedy	0.292±0.045
19/20	RxnMolDQN	Greedy	0.342±0.213
19/20	RxnMolDQN	Random Walk	0.294±0.046
21/22	RxnMolDQN	Epsilon Greedy	0.293±0.049
21/22	RxnMolDQN	Greedy	0.317±0.209
21/22	RxnMolDQN	Random Walk	0.295±0.047
23/24	RxnMolDQN	Epsilon Greedy	0.293±0.046
23/24	RxnMolDQN	Greedy	0.322±0.211
23/24	RxnMolDQN	Random Walk	0.293±0.046

Appendix H: Silly Walks

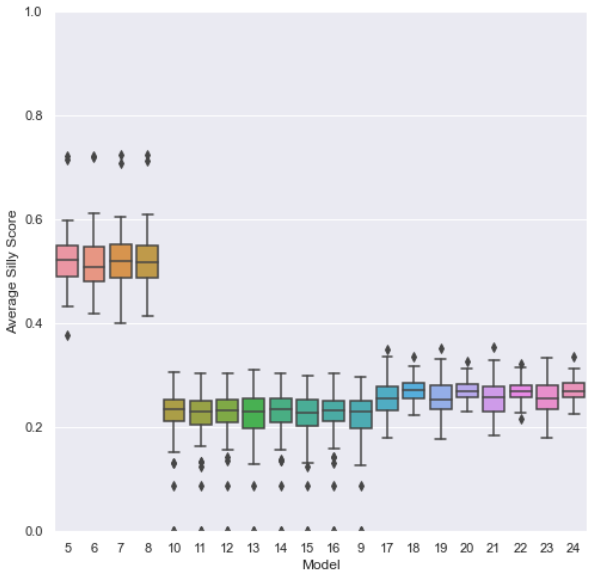


Figure H1: Average silly score for MolDQN (methods 3-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED.

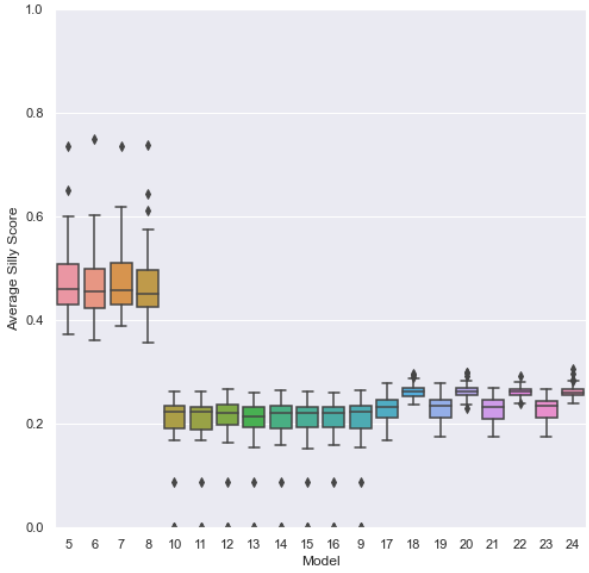


Figure H2: Average silly score for MolDQN (methods 3-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, penalized log.

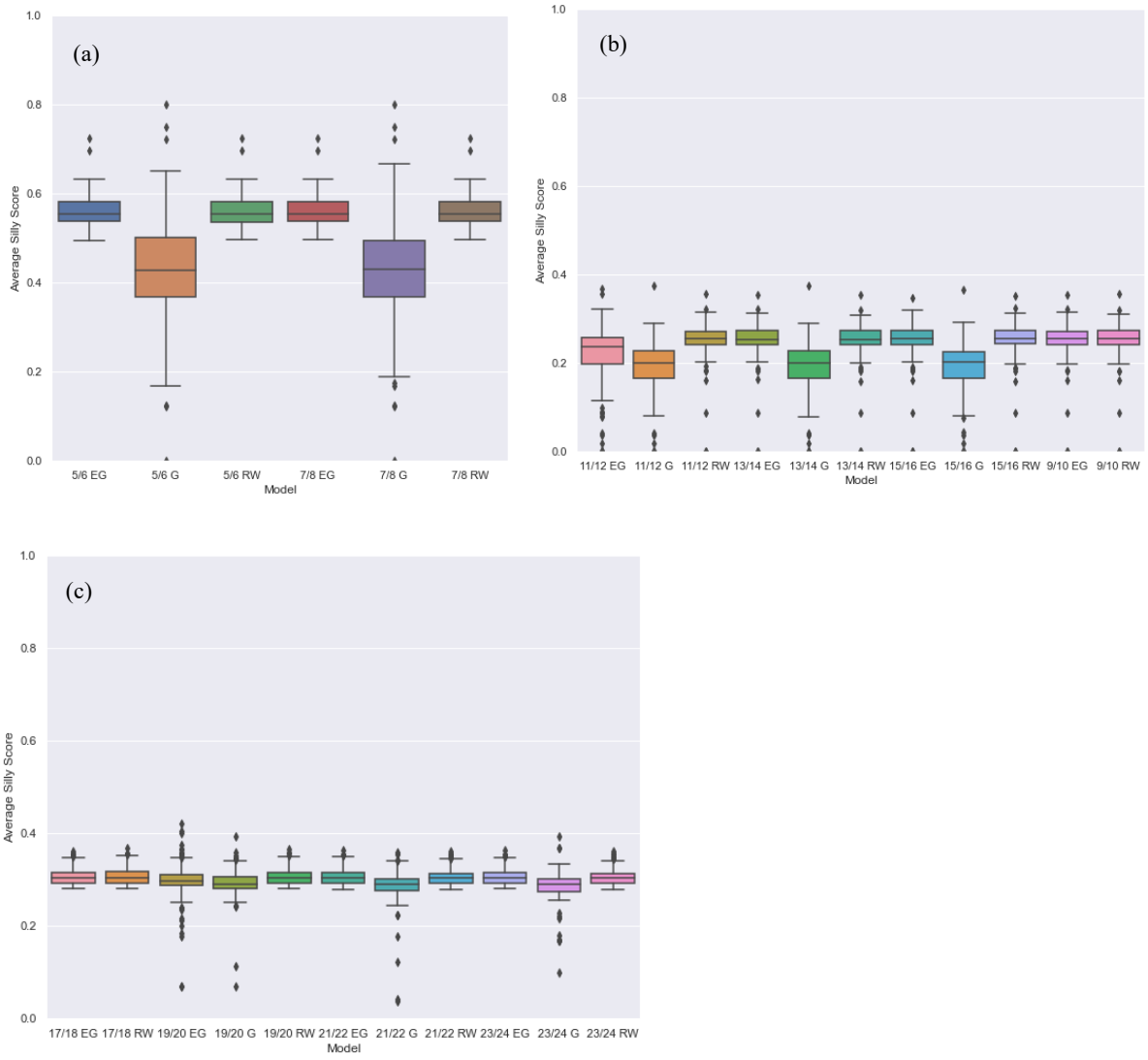


Figure H3: Average silly score for MolDQN (methods 3-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) baselines computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED. "E", "EG", and "RW" stand for epsilon, epsilon greedy, and random walk, respectively.

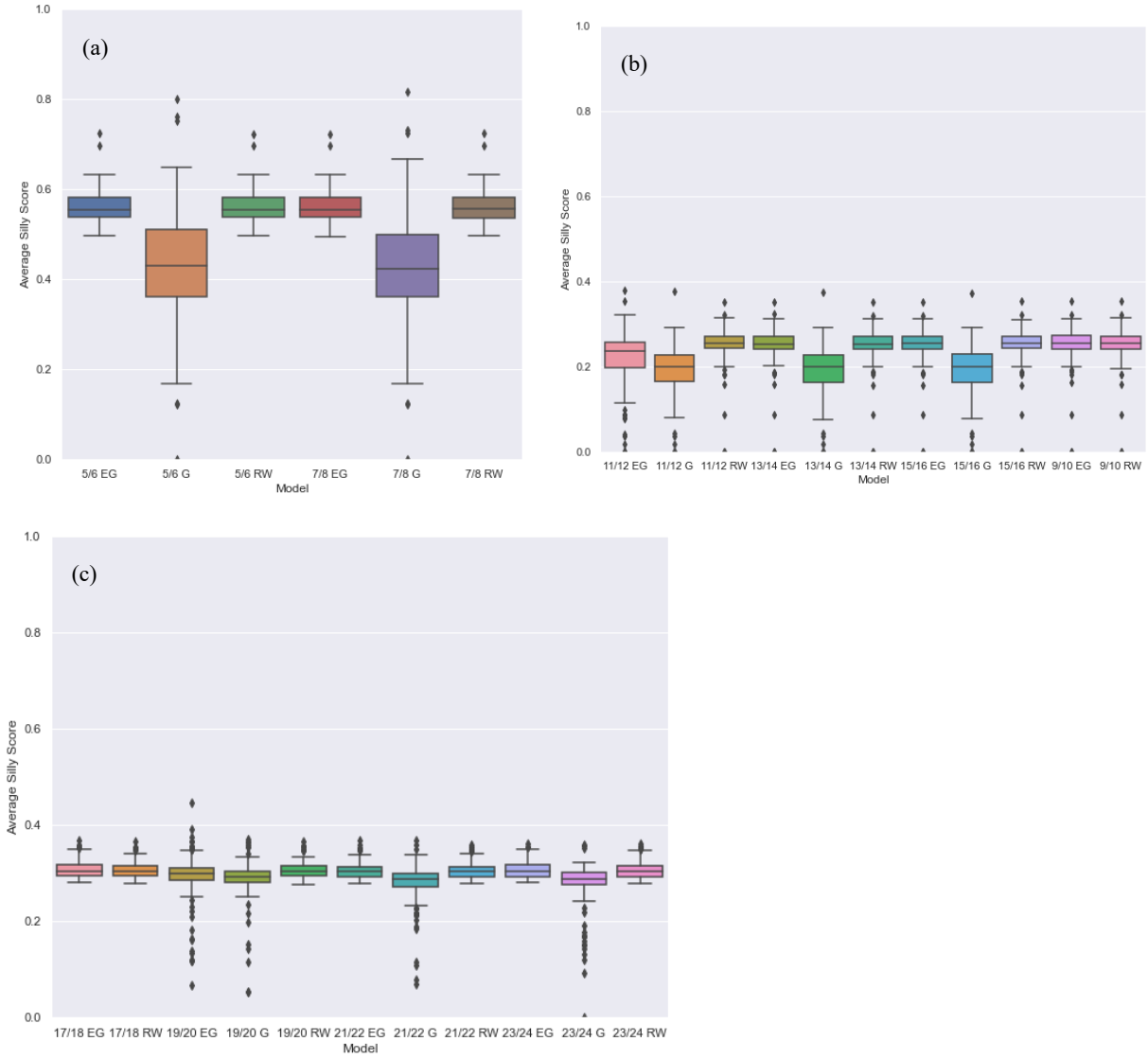


Figure H4: Average silly score for MolDQN (methods 3-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) baselines computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, penalized logP. "E", "EG", and "RW" stand for epsilon, epsilon greedy, and random walk, respectively.

Appendix I: Rediscovery

Table I1: Rediscovery for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, QED.

Method Number	Method	Baseline/DQN Methods	Success Rediscovery (/100)	Fail Rediscovery (/100)
5	MolDQN	Naive	96	4
6	MolDQN	Bootstrap	99	1
7	MolDQN	Naive	100	0
8	MolDQN	Bootstrap	97	3
5/6	MolDQN	Epsilon Greedy	100	0
5/6	MolDQN	Greedy	99	1
5/6	MolDQN	Random Walk	100	0
7/8	MolDQN	Epsilon Greedy	100	0
7/8	MolDQN	Greedy	99	1
7/8	MolDQN	Random Walk	100	0
9	RxnDQN	Naive	0	100
10	RxnDQN	Bootstrap	0	100
11	RxnDQN	Naive	0	100
12	RxnDQN	Bootstrap	0	100
13	RxnDQN	Naive	0	100
14	RxnDQN	Bootstrap	0	100
15	RxnDQN	Naive	0	100
16	RxnDQN	Bootstrap	0	100
9/10	RxnDQN	Epsilon Greedy	0	100
9/10	RxnDQN	Greedy	6	94
9/10	RxnDQN	Random Walk	0	100
11/12	RxnDQN	Epsilon Greedy	0	100
11/12	RxnDQN	Greedy	6	94
11/12	RxnDQN	Random Walk	0	100
13/14	RxnDQN	Epsilon Greedy	0	100
13/14	RxnDQN	Greedy	6	94
13/14	RxnDQN	Random Walk	0	100
15/16	RxnDQN	Epsilon Greedy	0	100
15/16	RxnDQN	Greedy	6	94
15/16	RxnDQN	Random Walk	0	100
17	RxnMolDQN	Naive	1	99
18	RxnMolDQN	Bootstrap	7	93
19	RxnMolDQN	Naive	3	97
20	RxnMolDQN	Bootstrap	4	96
21	RxnMolDQN	Naive	1	99
22	RxnMolDQN	Bootstrap	4	96
23	RxnMolDQN	Naive	6	94
24	RxnMolDQN	Bootstrap	2	98
17/18	RxnMolDQN	Epsilon Greedy	61	39
17/18	RxnMolDQN	Greedy	96	4
17/18	RxnMolDQN	Random Walk	52	48
19/20	RxnMolDQN	Epsilon Greedy	56	44
19/20	RxnMolDQN	Greedy	93	7

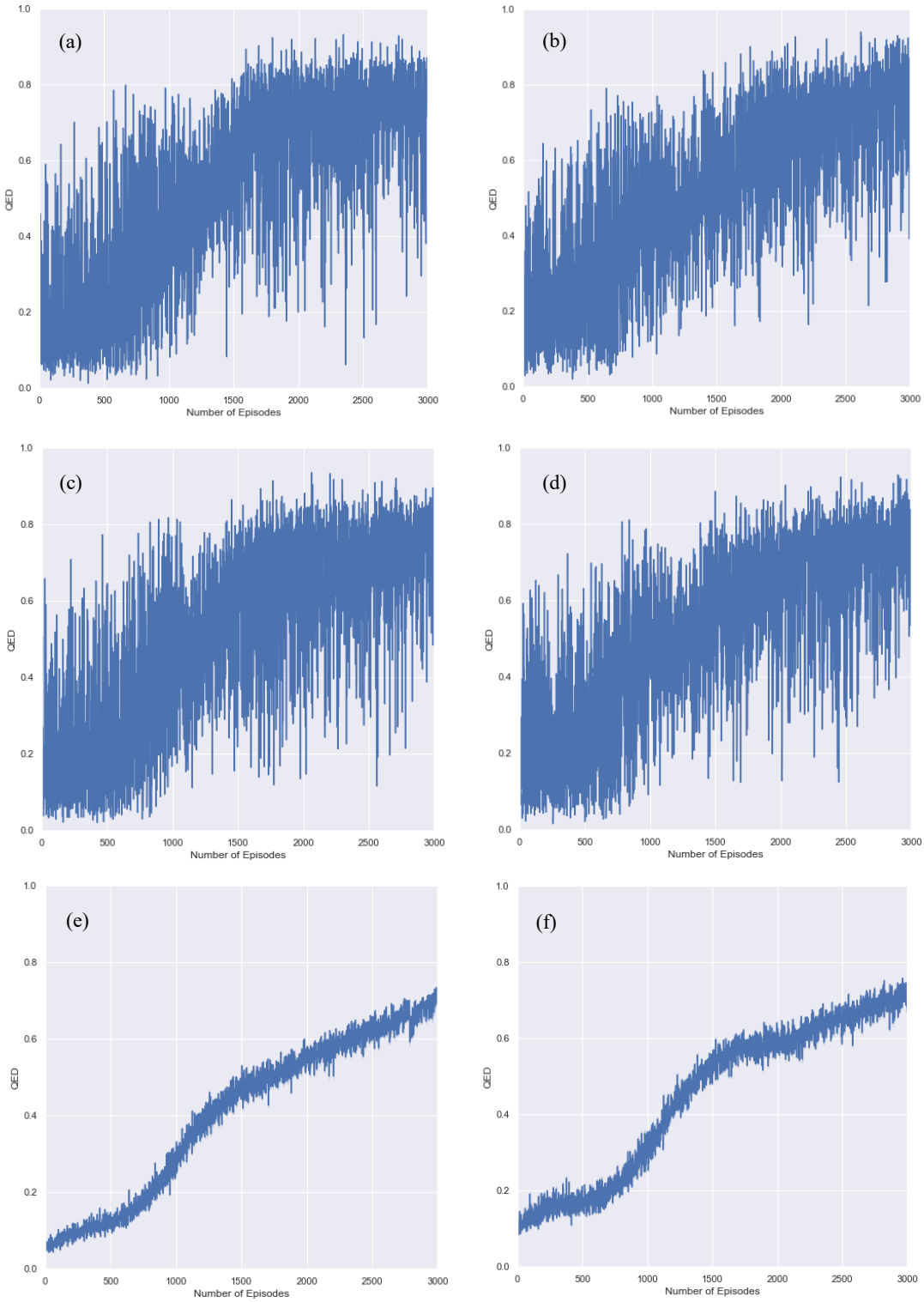
19/20	RxnMolDQN	Random Walk	60	40
21/22	RxnMolDQN	Epsilon Greedy	58	42
21/22	RxnMolDQN	Greedy	90	10
21/22	RxnMolDQN	Random Walk	64	36
23/24	RxnMolDQN	Epsilon Greedy	55	45
23/24	RxnMolDQN	Greedy	96	4
23/24	RxnMolDQN	Random Walk	51	49

Table I2: Rediscovery for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) using the reward function, penalized logP.

Method Number	Method	Baseline/DQN Methods	Success Rediscovery (/100)	Fail Rediscovery (/100)
5	MolDQN	Naive	100	0
6	MolDQN	Bootstrap	99	1
7	MolDQN	Naive	100	0
8	MolDQN	Bootstrap	99	1
5/6	MolDQN	Epsilon Greedy	100	0
5/6	MolDQN	Greedy	99	1
5/6	MolDQN	Random Walk	100	0
7/8	MolDQN	Epsilon Greedy	100	0
7/8	MolDQN	Greedy	99	1
7/8	MolDQN	Random Walk	100	0
9	RxnDQN	Naive	0	100
10	RxnDQN	Bootstrap	0	100
11	RxnDQN	Naive	0	100
12	RxnDQN	Bootstrap	0	100
13	RxnDQN	Naive	0	100
14	RxnDQN	Bootstrap	0	100
15	RxnDQN	Naive	0	100
16	RxnDQN	Bootstrap	0	100
9/10	RxnDQN	Epsilon Greedy	0	100
9/10	RxnDQN	Greedy	6	94
9/10	RxnDQN	Random Walk	0	100
11/12	RxnDQN	Epsilon Greedy	0	100
11/12	RxnDQN	Greedy	6	94
11/12	RxnDQN	Random Walk	0	100
13/14	RxnDQN	Epsilon Greedy	0	100
13/14	RxnDQN	Greedy	6	94
13/14	RxnDQN	Random Walk	0	100
15/16	RxnDQN	Epsilon Greedy	0	100
15/16	RxnDQN	Greedy	6	94
15/16	RxnDQN	Random Walk	0	100
17	RxnMolDQN	Naive	4	96
18	RxnMolDQN	Bootstrap	33	67
19	RxnMolDQN	Naive	4	96
20	RxnMolDQN	Bootstrap	29	71
21	RxnMolDQN	Naive	8	92
22	RxnMolDQN	Bootstrap	27	73
23	RxnMolDQN	Naive	7	93
24	RxnMolDQN	Bootstrap	21	79
17/18	RxnMolDQN	Epsilon Greedy	52	48
17/18	RxnMolDQN	Greedy	95	5
17/18	RxnMolDQN	Random Walk	55	45
19/20	RxnMolDQN	Epsilon Greedy	60	40
19/20	RxnMolDQN	Greedy	93	7
19/20	RxnMolDQN	Random Walk	53	47
21/22	RxnMolDQN	Epsilon Greedy	63	37

21/22	RxnMolDQN	Greedy	94	6
21/22	RxnMolDQN	Random Walk	51	49
23/24	RxnMolDQN	Epsilon Greedy	51	49
23/24	RxnMolDQN	Greedy	95	5
23/24	RxnMolDQN	Random Walk	62	38

Appendix J: Optimization Efficiency



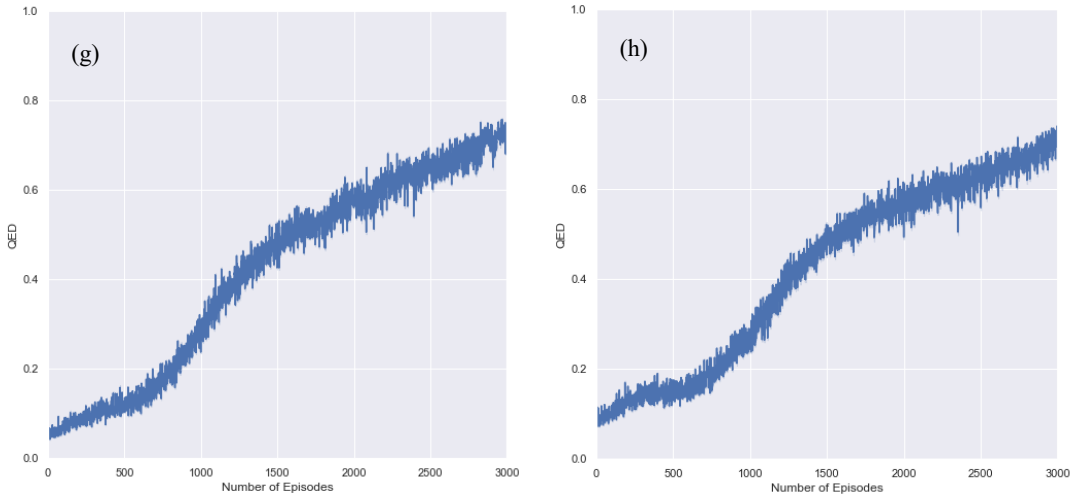
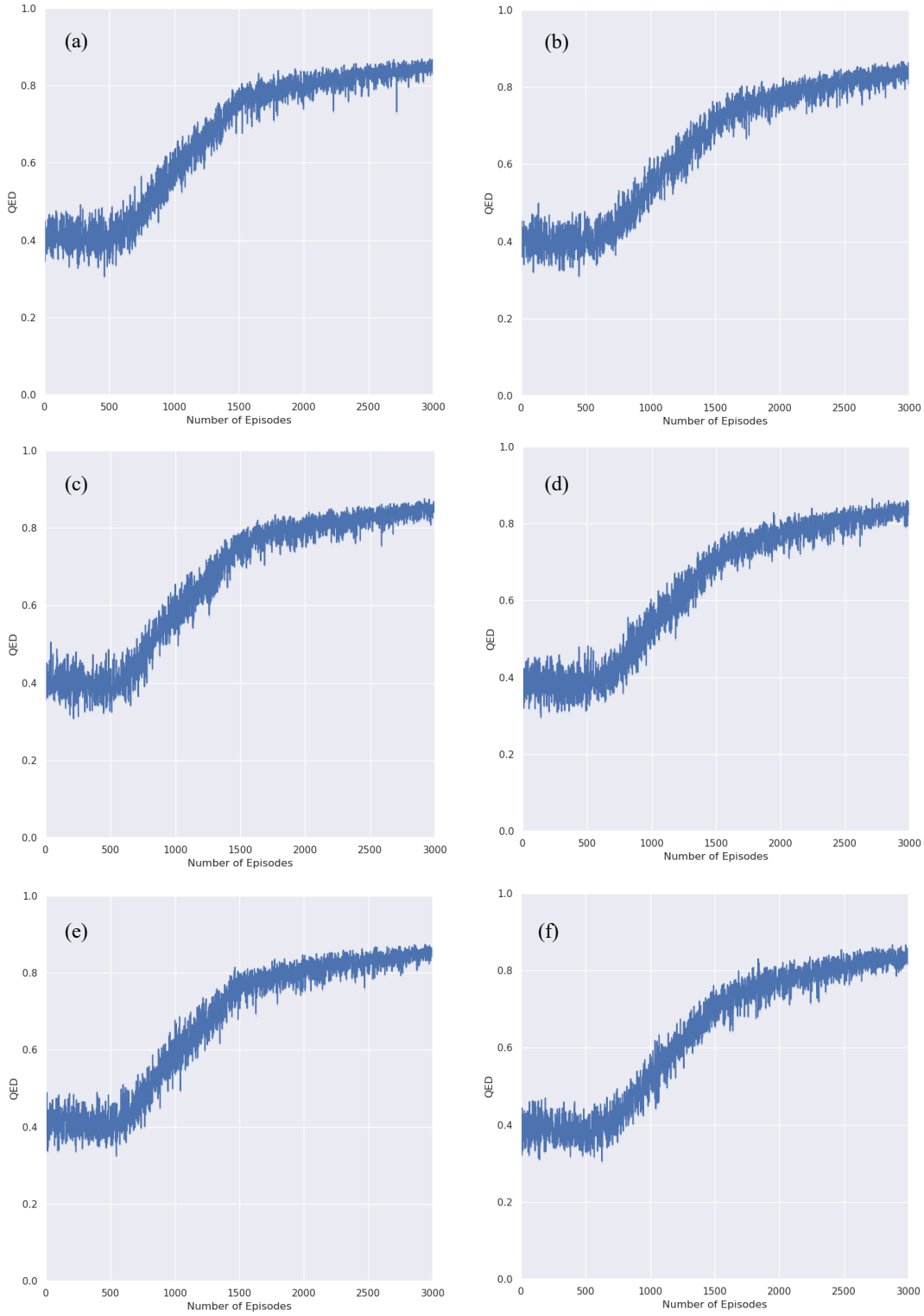


Figure J1: Reward progression for MolDQN (methods 1-8 in the order of (a) to (h)) using the reward function, QED.


(a)
(b)
(c)
(d)
(e)
(f)
QED
Number of Episodes
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000

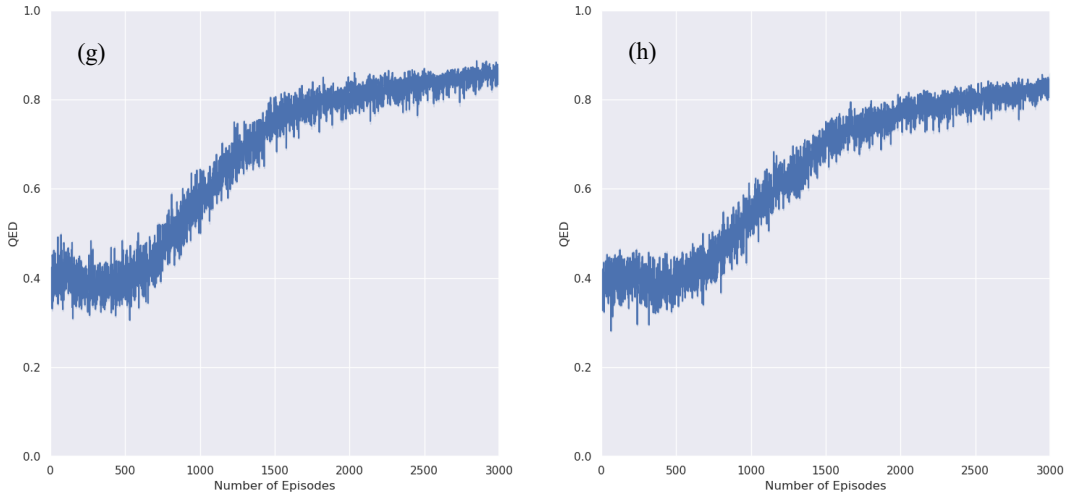
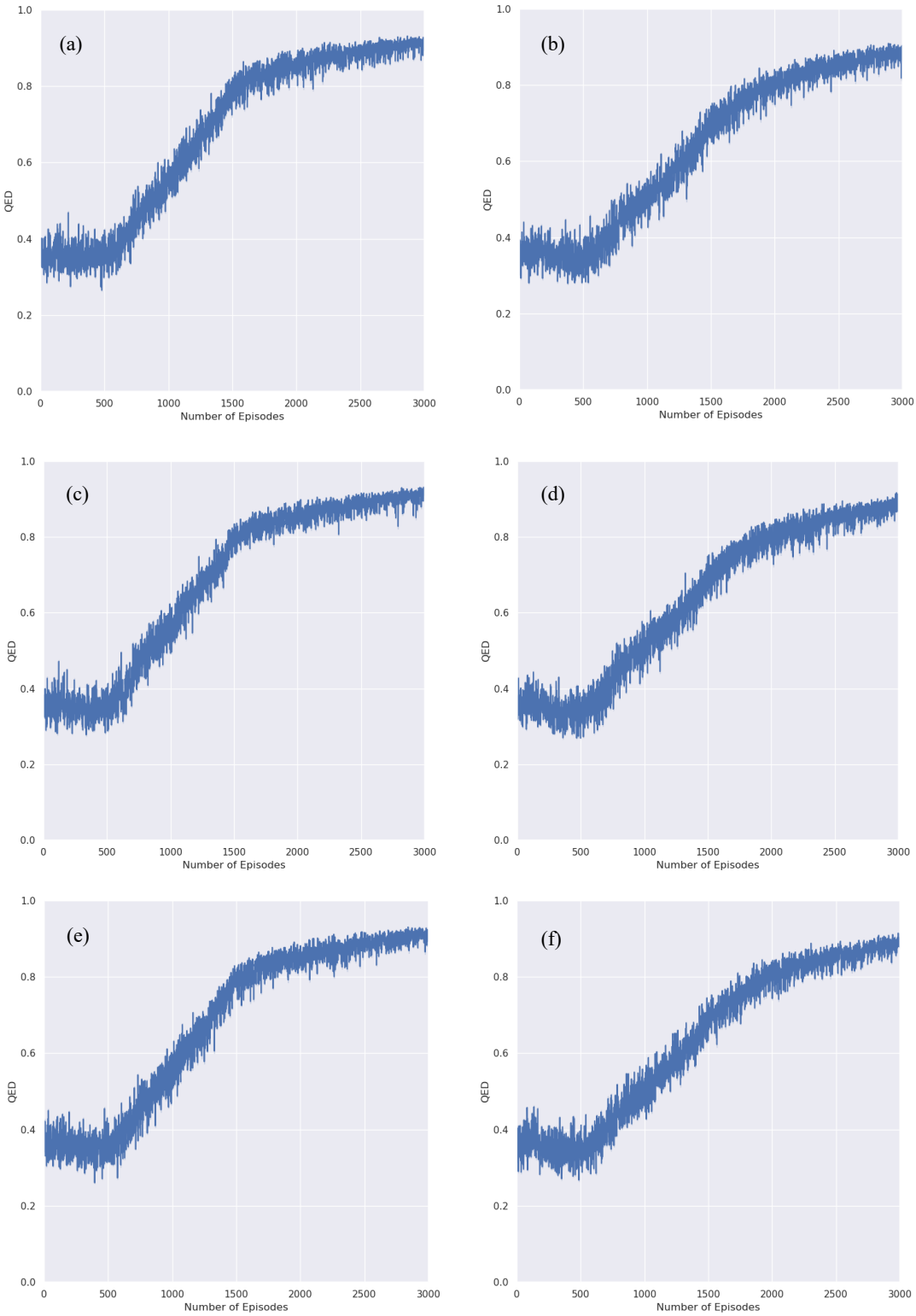


Figure J2: Reward progression for RxnDQN (methods 9-16 in the order of (a) to (h)) using the reward function, QED.


(a)
(b)
(c)
(d)
(e)
(f)
QED
Number of Episodes
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000

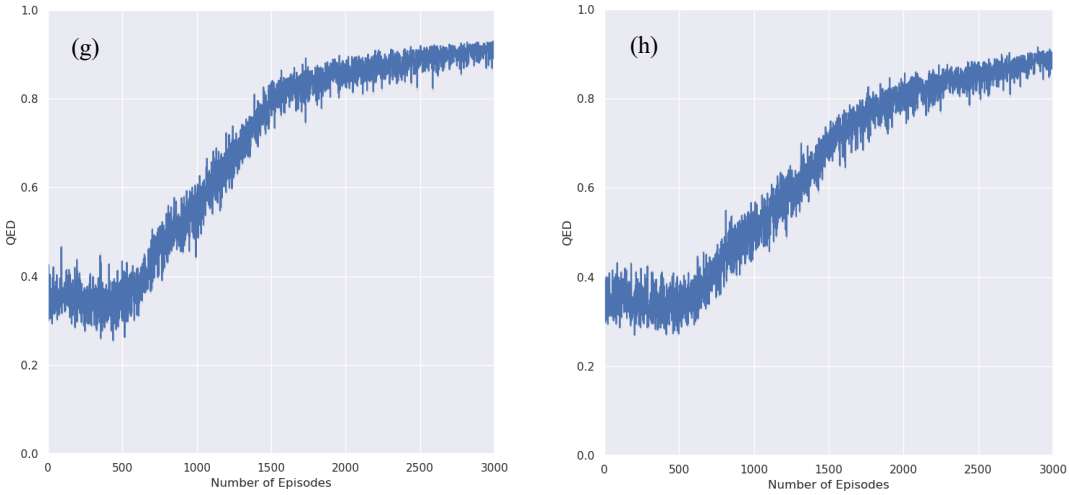
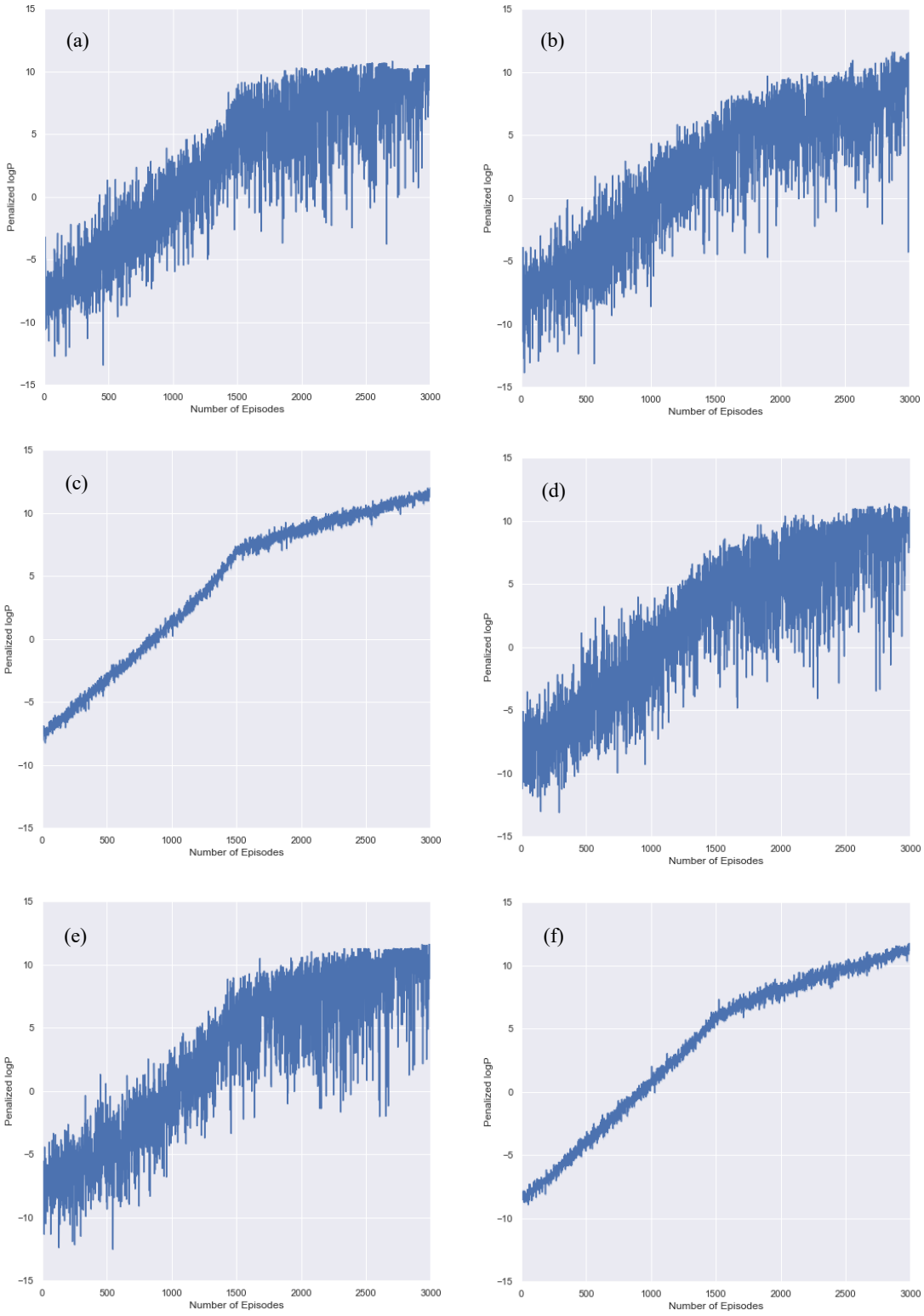


Figure J3: Reward progression for RxnMolDQN (methods 17-24 in the order of (a) to (h)) using the reward function, QED.


(a)
(b)
(c)
(d)
(e)
(f)
Penalized logP
Number of Episodes
15
10
5
0
−5
−10
−15
0
500
1000
1500
2000
2500
3000

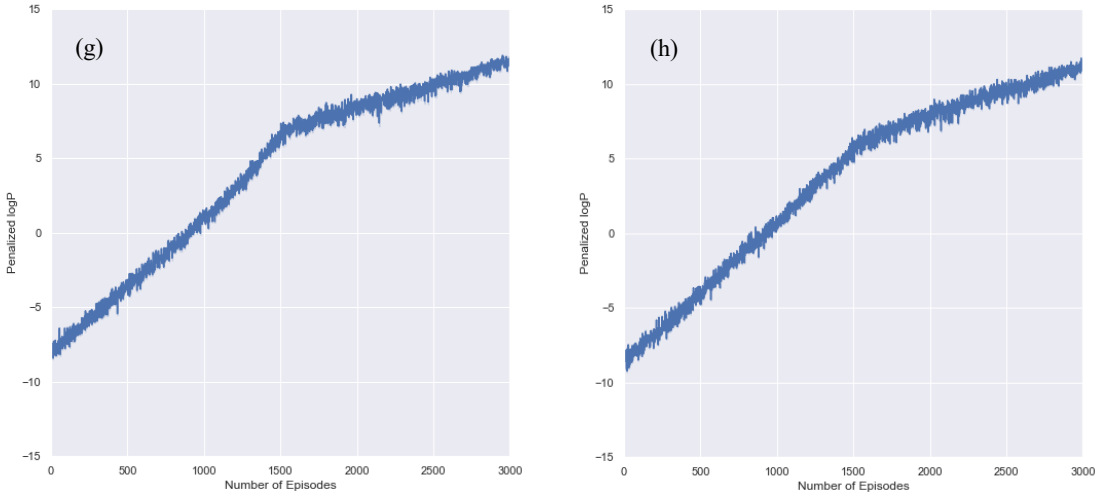
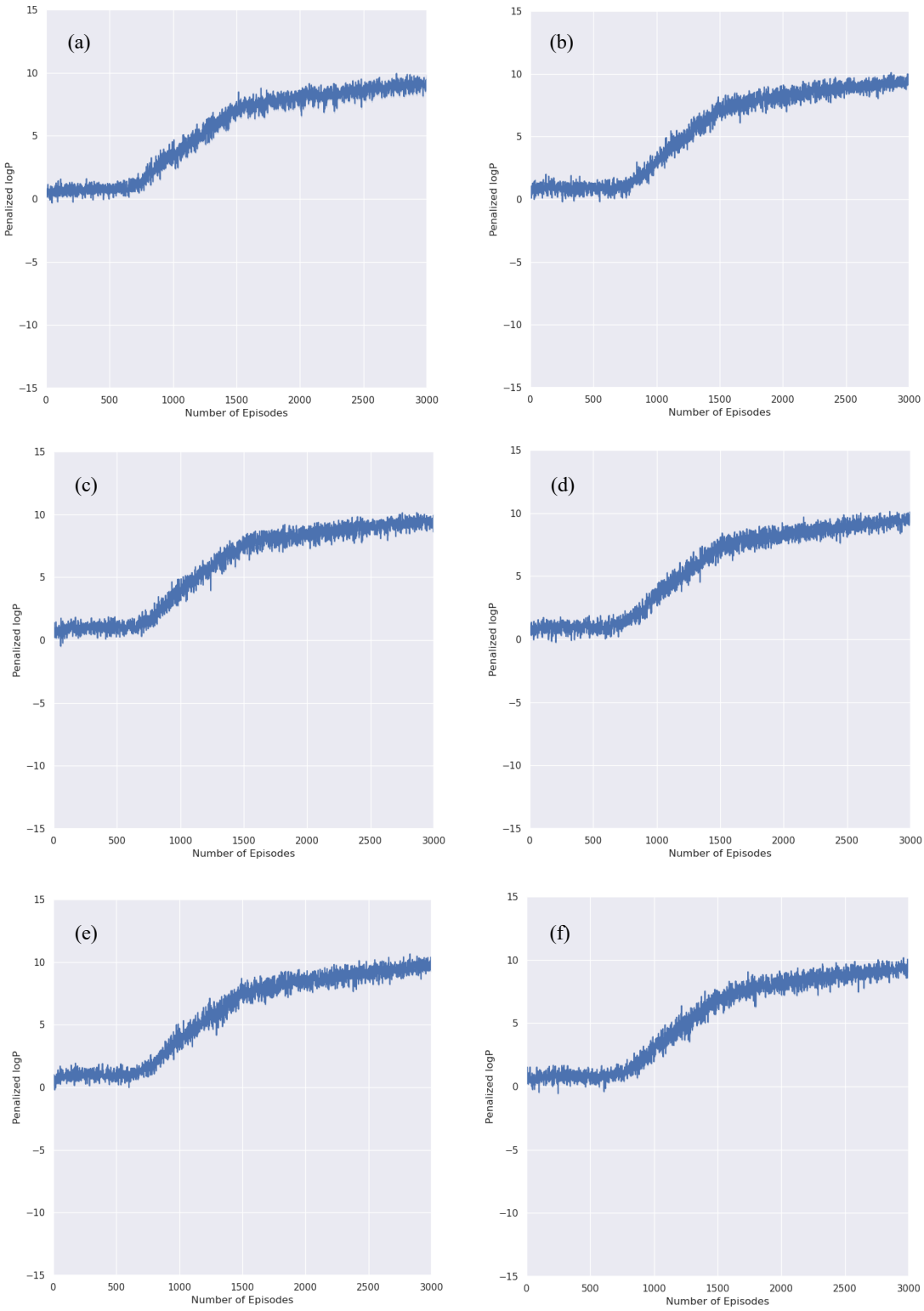


Figure J4: Reward progression for MolDQN (methods 1-8 in the order of (a) to (h)) using the reward function, penalized logP.


(a)
(b)
(c)
(d)
(e)
(f)
Penalized logP
Number of Episodes
15
10
5
0
−5
−10
−15
0
500
1000
1500
2000
2500
3000

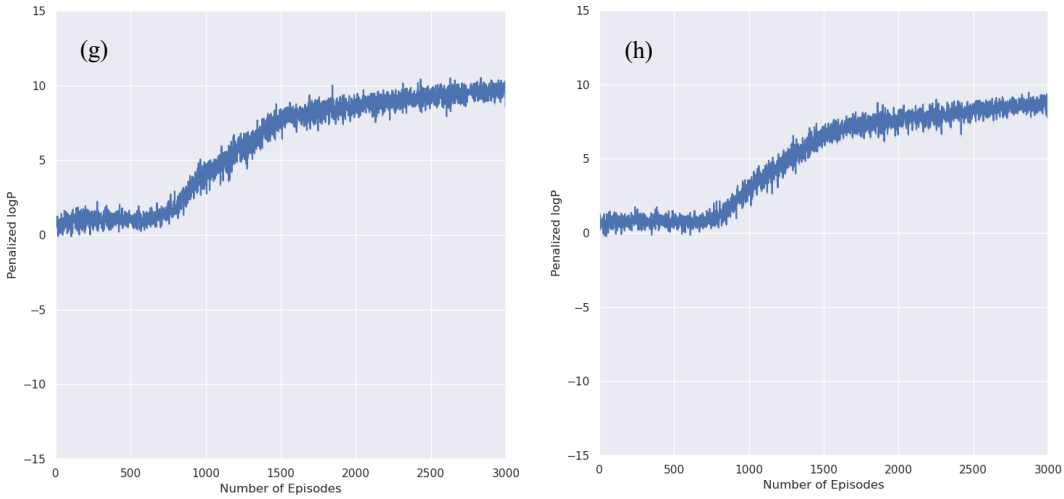
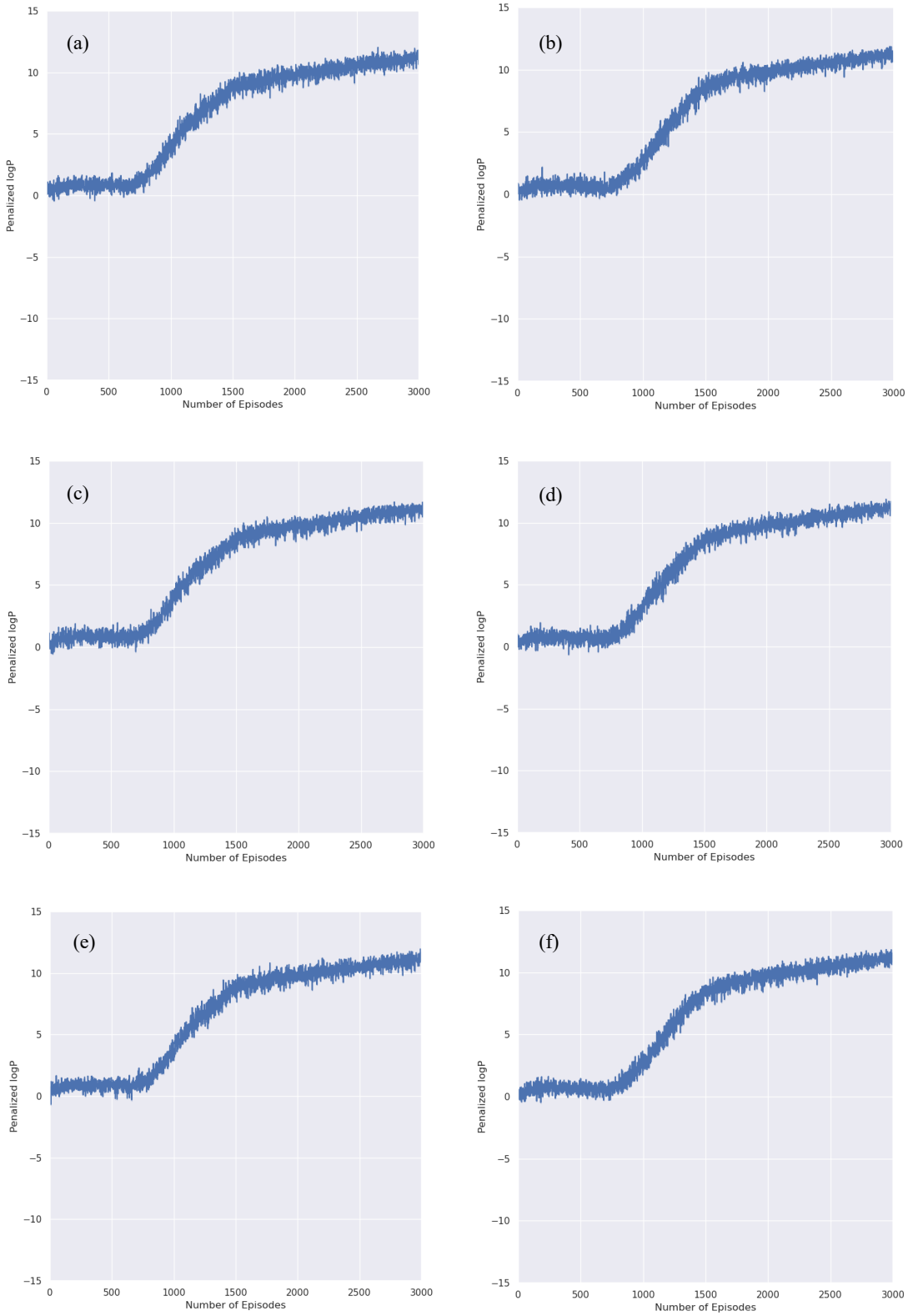


Figure J5: Reward progression for RxnDQN (methods 9-16 in the order of (a) to (h)) using the reward function, penalized logP.


(a)
(b)
(c)
(d)
(e)
(f)
Penalized logP
Number of Episodes
15
10
5
0
−5
−10
−15
0
500
1000
1500
2000
2500
3000

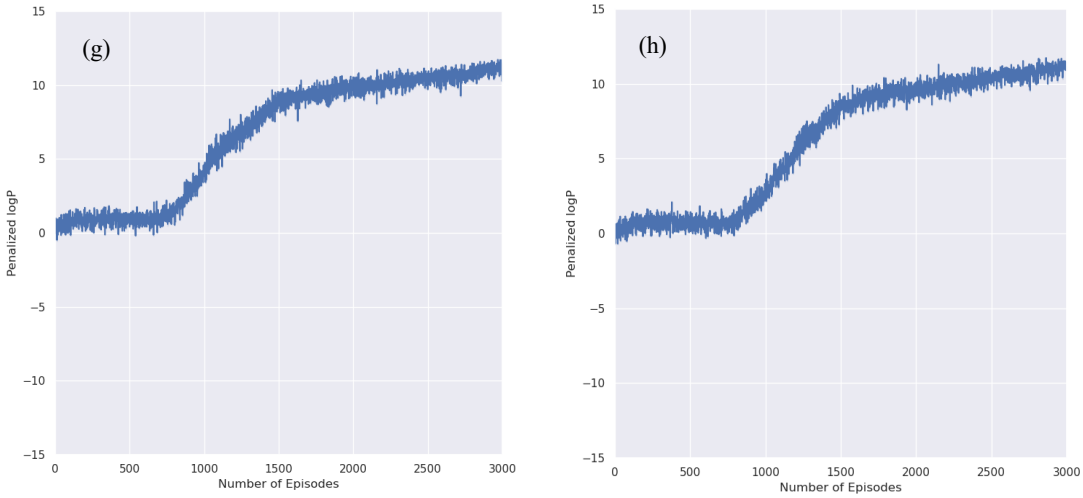


Figure J6: Reward progression for RxnMolDQN (methods 17-24 in the order of (a) to (h)) using the reward function, penalized logP.

Appendix K: Uniqueness

Table K1: Average percentage uniqueness of the molecules generated for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, QED. The values are approximated to one decimal place or significant figure.

Method Number	Baseline/DQN Methods	Average Uniqueness for Top 500 Episodes (%)	Average Uniqueness for Top 1,000 Episodes (%)	Average Uniqueness for Top 1,500 Episodes (%)	Average Uniqueness for Top 2,000 Episodes (%)	Average Uniqueness for Top 2,500 Episodes (%)	Average Uniqueness for Top 3,000 Episodes (%)
1	Naive	99.8	99.9	99.8	99.9	99.8	99.6
2	Bootstrap	100	100	99.9	99.9	99.9	99.9
3	Naive	100	100	100	100	100	99.9
4	Bootstrap	100	100	99.9	99.9	99.9	99.9
5	Naive	100	100	99.9±0.1	99.6±0.7	98.4±2.0	95.4±3.3
6	Bootstrap	100	100	99.9±0.1	99.8±0.5	99.0±1.6	96.7±3.2
7	Naive	100	100	99.9±0.1	99.6±0.7	98.3±1.7	95.3±2.9
8	Bootstrap	100	100	99.9±0.1	99.8±0.4	99.2±1.2	97.1±2.9
1/2	Epsilon Greedy	100	99.6	99.5	99.6	99.5	99.5
1/2	Greedy	0.2	0.1	0.07	0.05	0.04	0.03
1/2	Random Walk	100	99.9	99.9	99.7	99.7	99.6
3/4	Epsilon Greedy	100	100	99.9	99.8	99.7	99.5
3/4	Greedy	0.2	0.1	0.07	0.05	0.04	0.03
3/4	Random Walk	99.8	99.9	99.7	99.7	99.4	99.5
5/6	Epsilon Greedy	100	100	100	100	100	100
5/6	Greedy	0.2	0.1	0.07	0.05	0.04	0.03
5/6	Random Walk	100	100	100	100	100	100
7/8	Epsilon Greedy	100	100	100	100	100	100
7/8	Greedy	0.2	0.1	0.07	0.05	0.04	0.03
7/8	Random Walk	100	100	100	100	100	100
9	Naive	89.2±20.9	88.1±20.6	83.2±20.2	73.7±19.3	65.5±18.5	58.8±17.9
10	Bootstrap	89.0±20.9	88.1±20.6	84.6±20.3	77.3±19.7	70.1±19.2	63.7±18.7
11	Naive	89.2±20.9	88.2±20.6	82.9±20.1	73.2±19.2	65.0±18.3	58.2±17.6
12	Bootstrap	88.8±20.8	88.1±20.6	84.5±20.3	77.3±20.0	70.2±19.5	64.0±18.9
13	Naive	89.5±20.8	88.5±20.6	83.4±20.1	73.9±19.1	65.8±18.2	59.1±17.6
14	Bootstrap	89.4±20.8	88.7±20.6	85.5±20.3	78.5±19.7	71.2±19.1	64.9±18.6
15	Naive	89.6±20.8	88.6±20.6	83.7±20.1	74.3±19.2	66.1±18.3	59.4±17.7
16	Bootstrap	89.3±20.8	88.6±20.6	85.1±20.2	78.0±19.7	70.8±19.1	64.4±18.6
9/10	Epsilon Greedy	88.5±20.9	87.7±20.9	87.3±21.0	86.8±21.0	86.4±21.0	86.1±21.0
9/10	Random Walk	88.3±20.9	87.6±20.9	87.1±21.0	86.7±21.0	86.4±21.0	86.0±21.0
11/12	Epsilon Greedy	88.4±20.9	87.8±21.0	87.2±21.0	86.7±21.0	86.4±21.0	86.1±21.0
11/12	Greedy	52.1±30.2	45.6±28.8	41.9±27.7	39.5±26.9	37.6±26.2	36.1±25.6
11/12	Epsilon Greedy	52.0±30.3	45.7±28.8	42.1±27.8	39.5±26.8	37.6±26.1	36.0±25.6
11/12	Random Walk	88.3±20.9	87.6±20.9	87.1±21.0	86.7±21.0	86.3±21.0	86.0±21.0
13/14	Epsilon Greedy	88.6±20.7	88.2±20.7	87.7±20.7	87.3±20.7	87.0±20.6	86.7±20.6
13/14	Greedy	52.9±30.5	46.5±29.1	42.9±28.0	40.4±27.2	38.5±26.4	37.0±25.8
13/14	Random Walk	88.7±20.8	88.2±20.7	87.7±20.7	87.3±20.7	87.0±20.7	86.7±20.7
15/16	Epsilon Greedy	88.8±20.8	88.2±20.7	87.9±20.7	87.5±20.7	87.1±20.7	86.9±20.7
15/16	Greedy	52.9±30.5	46.6±29.1	43.0±28.1	40.5±27.2	38.6±26.4	37.0±25.8
15/16	Random Walk	88.8±20.8	88.2±20.7	87.8±20.7	87.5±20.7	87.1±20.7	86.8±20.7
17	Naive	99.2±0.7	98.2±1.0	93.1±3.1	81.2±5.3	70.6±5.9	61.5±6.2
18	Bootstrap	99.7±0.4	99.4±0.4	98.5±0.7	93.2±3.0	85.5±4.6	78.1±5.6
19	Naive	99.1±0.7	98.2±1.0	93.5±3.2	81.7±5.4	70.9±6.1	61.8±6.3
20	Bootstrap	99.7±0.3	99.4±0.4	98.6±0.6	93.4±2.8	85.9±4.4	78.4±5.3
21	Naive	99.2±0.7	98.4±0.8	93.7±2.8	81.7±5.1	70.9±5.9	62.0±6.1
22	Bootstrap	99.7±0.3	99.4±0.3	98.6±0.7	93.4±3.0	85.9±4.7	78.5±5.9

23	Naive	99.2±0.7	98.3±0.8	93.5±2.8	81.7±5.1	71.0±5.8	61.9±6.0
24	Bootstrap	99.7±0.4	99.4±0.4	98.7±0.8	94.0±3.1	86.3±4.4	78.6±5.3
17/18	Epsilon Greedy	99.7±0.4	99.6±0.4	99.5±0.5	99.4±0.5	99.3±0.5	99.3±0.5
17/18	Random Walk	99.7±0.4	99.6±0.4	99.4±0.5	99.4±0.5	99.3±0.6	99.2±0.6
19/20	Epsilon Greedy	99.7±0.4	99.6±0.4	99.5±0.4	99.4±0.5	99.3±0.5	99.2±0.6
19/20	Greedy	74.7±27.3	71.9±26.8	70.2±26.4	68.9±26.2	68.0±26.0	67.1±25.8
19/20	Epsilon Greedy	70.7±30.6	68.2±29.8	66.7±29.3	65.6±28.9	64.7±28.5	63.9±28.2
19/20	Random Walk	99.8±0.3	99.6±0.3	99.5±0.4	99.4±0.5	99.3±0.5	99.2±0.5
21/22	Epsilon Greedy	99.8±0.3	99.6±0.3	99.5±0.4	99.4±0.5	99.3±0.5	99.2±0.6
21/22	Greedy	76.1±26.3	73.6±25.8	72.0±25.4	70.8±25.1	69.8±24.8	68.9±24.6
21/22	Random Walk	99.8±0.3	99.6±0.5	99.5±0.5	99.4±0.6	99.3±0.6	99.2±0.6
23/24	Epsilon Greedy	99.8±0.3	99.6±0.4	99.5±0.5	99.4±0.5	99.3±0.5	99.2±0.5
23/24	Greedy	74.4±27.7	71.6±27.0	69.9±26.6	68.7±26.3	67.9±26.0	67.1±25.8
23/24	Random Walk	99.7±0.4	99.6±0.4	99.5±0.5	99.4±0.6	99.3±0.6	99.2±0.6

Table K2: Average percentage uniqueness of the molecules generated for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, penalized logP. The values are approximated to one decimal place or significant figure.

Method Number	Baseline/DQN Methods	Average Uniqueness for Top 500 Episodes (%)	Average Uniqueness for Top 1,000 Episodes (%)	Average Uniqueness for Top 1,500 Episodes (%)	Average Uniqueness for Top 2,000 Episodes (%)	Average Uniqueness for Top 2,500 Episodes (%)	Average Uniqueness for Top 3,000 Episodes (%)
1	Naive	100	99.8	99.6	98.6	95.0	88.5
2	Bootstrap	100	100	99.6	99.1	98.2	97.1
3	Naive	99.8	99.9	99.4	98.7	95.8	91.2
4	Bootstrap	100	100	99.4	98.1	96.7	93.6
5	Naive	100	99.9±0.1	99.8±0.3	99.0±1.6	97.7±3.0	94.6±4.4
6	Bootstrap	100	99.9±0.1	99.9±0.1	99.5±0.5	98.6±1.7	96.9±3.3
7	Naive	100	99.9±0.1	99.8±0.2	99.0±1.4	97.9±2.7	94.8±3.8
8	Bootstrap	100	99.9±0.1	99.9±0.3	99.3±1.6	98.4±3.1	96.7±4.6
1/2	Epsilon Greedy	100	100	99.8	99.7	99.7	99.6
1/2	Greedy	0.2	0.1	0.07	0.05	0.04	0.03
1/2	Random Walk	99.8	99.8	99.8	99.8	99.7	99.7
3/4	Epsilon Greedy	99.6	99.7	99.5	99.5	99.2	99.3
3/4	Greedy	0.2	0.1	0.07	0.05	0.04	0.03
3/4	Random Walk	100	99.9	99.8	99.7	99.7	99.7
5/6	Epsilon Greedy	100	100	100	100	100	100
5/6	Greedy	0.2	0.1	0.07	0.05	0.04	0.03
5/6	Random Walk	100	100	100	100	100	100
7/8	Epsilon Greedy	100	100	100	100	100	100
7/8	Greedy	0.2	0.1	0.07	0.05	0.04	0.03
7/8	Random Walk	100	100	100	100	100	100
9	Naive	89.7±20.9	89.6±20.8	88.7±20.7	85.7±20.6	82.3±20.7	88.7±20.7
10	Bootstrap	89.2±20.9	89.0±20.7	88.9±20.8	86.5±20.6	83.5±20.6	88.9±20.8
11	Naive	89.7±20.9	89.5±20.8	88.7±20.8	85.4±20.6	82.0±20.7	88.7±20.8
12	Bootstrap	89.1±20.9	88.9±20.7	89.0±20.7	87.0±20.6	84.2±20.6	89.0±20.7
13	Naive	89.8±20.8	89.8±20.8	89.2±20.8	86.2±20.7	82.9±20.7	89.2±20.8
14	Bootstrap	89.6±20.8	89.5±20.7	89.4±20.8	87.3±20.6	84.5±20.7	89.4±20.8
15	Naive	89.8±20.9	89.9±20.8	89.4±20.8	86.7±20.6	83.6±20.7	89.4±20.8
16	Bootstrap	89.4±20.8	89.5±20.7	89.6±20.8	87.4±20.6	84.5±20.7	89.6±20.8
9/10	Epsilon Greedy	88.2±20.9	87.5±20.9	87.0±20.9	86.6±20.9	86.2±20.9	87.0±20.9
9/10	Random Walk	88.3±20.9	87.6±21.0	87.1±21.0	86.7±21.0	86.3±21.0	87.1±21.0
11/12	Epsilon Greedy	88.6±20.9	87.7±20.9	87.2±20.9	86.8±20.9	86.4±21.0	87.2±21.0
11/12	Greedy	52.0±30.3	45.6±28.8	41.9±27.7	39.4±26.9	37.5±26.2	41.9±27.7
11/12	Epsilon Greedy	52.0±30.4	45.6±28.9	42.0±27.9	39.6±27.1	37.7±26.3	42.0±27.9
11/12	Random Walk	88.5±20.9	87.7±20.9	87.2±20.9	86.8±21.0	86.4±21.0	87.2±20.9
13/14	Epsilon Greedy	88.6±20.8	88.1±20.7	87.7±20.7	87.3±20.7	87.0±20.7	87.7±20.7
13/14	Greedy	52.8±30.3	46.5±29.0	42.9±28.0	40.4±27.1	38.5±26.5	42.9±28.0
13/14	Random Walk	88.5±20.8	88.1±20.7	87.7±20.7	87.4±20.7	87.1±20.7	87.7±20.7
15/16	Epsilon Greedy	88.7±20.8	88.1±20.7	87.7±20.7	87.4±20.7	87.0±20.7	87.7±20.7
15/16	Greedy	53.1±30.4	46.6±29.1	43.1±28.1	40.5±27.2	38.5±26.4	43.1±28.1
15/16	Random Walk	88.5±20.7	88.0±20.7	87.7±20.7	87.3±20.7	87.1±20.7	87.7±20.7
17	Naive	99.3±0.8	98.6±0.8	97.2±1.8	93.5±4.2	89.4±6.5	85.5±8.5

18	Bootstrap	99.7±0.3	99.6±0.3	99.5±0.3	99.3±0.6	98.7±1.1	98.0±1.6
19	Naive	99.2±0.8	98.6±1.0	97.3±1.9	93.5±4.3	89.5±6.5	85.7±8.4
20	Bootstrap	99.8±0.3	99.6±0.3	99.5±0.3	99.2±0.6	98.6±1.1	97.9±1.8
21	Naive	99.3±0.7	98.7±0.9	97.6±1.6	94.4±3.7	90.7±6.1	87.0±8.2
22	Bootstrap	99.7±0.3	99.5±0.3	99.5±0.3	99.2±0.6	98.7±1.2	98.0±1.9
23	Naive	99.3±0.7	98.8±0.7	97.7±1.5	94.5±3.6	90.9±6.0	87.1±8.2
24	Bootstrap	99.8±0.3	99.6±0.3	99.5±0.3	99.2±0.5	98.8±0.9	98.0±1.6
17/18	Epsilon Greedy	99.8±0.3	99.6±0.4	99.5±0.4	99.4±0.5	99.3±0.5	99.2±0.5
17/18	Random Walk	99.8±0.3	99.6±0.4	99.5±0.5	99.4±0.5	99.3±0.6	99.2±0.6
19/20	Epsilon Greedy	99.7±0.4	99.6±0.4	99.5±0.5	99.4±0.5	99.3±0.5	99.2±0.6
19/20	Greedy	70.5±30.9	67.9±30.1	66.3±29.6	65.1±29.2	64.2±28.8	63.5±28.6
19/20	Epsilon Greedy	67.2±34.6	65.0±33.6	63.6±33.0	62.6±32.6	61.8±32.1	61.1±31.8
19/20	Random Walk	99.7±0.4	99.6±0.4	99.5±0.5	99.4±0.6	99.3±0.6	99.3±0.6
21/22	Epsilon Greedy	99.8±0.3	99.7±0.3	99.5±0.4	99.4±0.5	99.3±0.5	99.2±0.5
21/22	Greedy	68.3±33.0	66.0±32.1	64.8±31.6	63.8±31.2	63.0±30.8	62.2±30.5
21/22	Random Walk	99.7±0.5	99.6±0.4	99.5±0.5	99.4±0.5	99.3±0.5	99.2±0.6
23/24	Epsilon Greedy	99.7±0.4	99.6±0.5	99.4±0.5	99.4±0.6	99.3±0.6	99.2±0.6
23/24	Greedy	70.0±32.6	67.7±31.7	66.2±31.2	65.1±30.7	64.3±30.4	63.6±30.2
23/24	Random Walk	99.7±0.4	99.6±0.4	99.5±0.5	99.4±0.5	99.3±0.5	99.2±0.6

Appendix L: Computational Efficiency

Table L1: Computational efficiency for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, QED. The values are approximated to one decimal place.

Method Number	Method	Baseline/DQN Methods	Average Time (hr)
1	MolDQN	Naive	17.3
2	MolDQN	Bootstrap	17.2
3	MolDQN	Naive	18.5
4	MolDQN	Bootstrap	18.9
5	MolDQN	Naive	23.7±8.8
6	MolDQN	Bootstrap	24.9±13.8
7	MolDQN	Naive	28.3±6.6
8	MolDQN	Bootstrap	26.8±7.3
1/2	MolDQN	Epsilon Greedy	0.4
1/2	MolDQN	Greedy	0.2
1/2	MolDQN	Random Walk	0.3
3/4	MolDQN	Epsilon Greedy	1.6
3/4	MolDQN	Greedy	0.2
3/4	MolDQN	Random Walk	0.3
5/6	MolDQN	Epsilon Greedy	3.9±2.8
5/6	MolDQN	Greedy	5.4±2.7
5/6	MolDQN	Random Walk	2.3±1.5
7/8	MolDQN	Epsilon Greedy	3.9±3.0
7/8	MolDQN	Greedy	6.1±3.2
7/8	MolDQN	Random Walk	2.4±1.5
9	RxnDQN	Naive	13.5±4.8
10	RxnDQN	Bootstrap	13.6±4.9
11	RxnDQN	Naive	13.4±4.8
12	RxnDQN	Bootstrap	13.6±4.8
13	RxnDQN	Naive	13.3±4.9
14	RxnDQN	Bootstrap	13.6±4.9
15	RxnDQN	Naive	13.1±4.8
16	RxnDQN	Bootstrap	13.4±4.7
9/10	RxnDQN	Epsilon Greedy	9.7±3.4
9/10	RxnDQN	Greedy	7.8±3.3
9/10	RxnDQN	Random Walk	9.2±3.2
11/12	RxnDQN	Epsilon Greedy	9.6±3.4
11/12	RxnDQN	Greedy	7.7±3.3
11/12	RxnDQN	Random Walk	9.4±3.4
13/14	RxnDQN	Epsilon Greedy	9.5±3.4
13/14	RxnDQN	Greedy	7.6±3.2
13/14	RxnDQN	Random Walk	9.1±3.2
15/16	RxnDQN	Epsilon Greedy	9.6±3.4
15/16	RxnDQN	Greedy	7.7±3.3
15/16	RxnDQN	Random Walk	9.1±3.2
17	RxnMolDQN	Naive	14.1±4.6

18	RxnMolDQN	Bootstrap	28.5±5.2
19	RxnMolDQN	Naive	13.9±4.3
20	RxnMolDQN	Bootstrap	28.8±5.3
21	RxnMolDQN	Naive	14.8±4.6
22	RxnMolDQN	Bootstrap	29.7±5.6
23	RxnMolDQN	Naive	14.8±5.4
24	RxnMolDQN	Bootstrap	29.5±5.2
17/18	RxnMolDQN	Epsilon Greedy	19.0±4.1
17/18	RxnMolDQN	Greedy	17.5±5.9
17/18	RxnMolDQN	Random Walk	18.0±3.8
19/20	RxnMolDQN	Epsilon Greedy	19.1±4.0
19/20	RxnMolDQN	Greedy	18.1±5.4
19/20	RxnMolDQN	Random Walk	18.2±3.8
21/22	RxnMolDQN	Epsilon Greedy	18.9±4.1
21/22	RxnMolDQN	Greedy	18.0±5.1
21/22	RxnMolDQN	Random Walk	17.6±3.8
23/24	RxnMolDQN	Epsilon Greedy	18.7±4.0
23/24	RxnMolDQN	Greedy	17.5±5.7
23/24	RxnMolDQN	Random Walk	17.6±3.9

Table L2: Computational efficiency for MolDQN (methods 1-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed using 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) or an empty molecule using the reward function, penalized logP. The values are approximated to one decimal place.

Method Number	Method	Baseline/DQN Methods	Average Time (hr)
1	MolDQN	Naive	18.9
2	MolDQN	Bootstrap	21.8
3	MolDQN	Naive	19.5
4	MolDQN	Bootstrap	22.6
5	MolDQN	Naive	28.4±15.3
6	MolDQN	Bootstrap	31.1±14.0
7	MolDQN	Naive	29.2±8.9
8	MolDQN	Bootstrap	33.5±9.4
1/2	MolDQN	Epsilon Greedy	0.4
1/2	MolDQN	Greedy	0.2
1/2	MolDQN	Random Walk	0.3
3/4	MolDQN	Epsilon Greedy	0.4
3/4	MolDQN	Greedy	0.2
3/4	MolDQN	Random Walk	0.3
5/6	MolDQN	Epsilon Greedy	3.6±2.3
5/6	MolDQN	Greedy	7.4±7.6
5/6	MolDQN	Random Walk	3.4±4.1
7/8	MolDQN	Epsilon Greedy	3.8±2.4
7/8	MolDQN	Greedy	5.6±3.0
7/8	MolDQN	Random Walk	2.4±1.6
9	RxnDQN	Naive	14.6±4.8
10	RxnDQN	Bootstrap	14.4±4.7
11	RxnDQN	Naive	14.4±4.6
12	RxnDQN	Bootstrap	14.6±5.0
13	RxnDQN	Naive	14.5±4.7
14	RxnDQN	Bootstrap	14.6±4.8
15	RxnDQN	Naive	14.5±4.6
16	RxnDQN	Bootstrap	14.6±4.8
9/10	RxnDQN	Epsilon Greedy	9.8±3.5
9/10	RxnDQN	Greedy	7.7±3.2
9/10	RxnDQN	Random Walk	9.2±3.1
11/12	RxnDQN	Epsilon Greedy	9.7±3.4
11/12	RxnDQN	Greedy	7.7±3.2
11/12	RxnDQN	Random Walk	9.2±3.2
13/14	RxnDQN	Epsilon Greedy	9.6±3.4
13/14	RxnDQN	Greedy	7.6±3.2
13/14	RxnDQN	Random Walk	9.1±3.2
15/16	RxnDQN	Epsilon Greedy	9.6±3.4
15/16	RxnDQN	Greedy	7.7±3.3
15/16	RxnDQN	Random Walk	9.2±3.1
17	RxnMolDQN	Naive	16.5±4.1
18	RxnMolDQN	Bootstrap	34.9±5.7
19	RxnMolDQN	Naive	16.4±4.3
20	RxnMolDQN	Bootstrap	34.9±5.5
21	RxnMolDQN	Naive	17.3±4.2

22	RxnMolDQN	Bootstrap	36.0±6.2
23	RxnMolDQN	Naive	17.2±4.2
24	RxnMolDQN	Bootstrap	39.5±16.4
17/18	RxnMolDQN	Epsilon Greedy	18.9±4.0
17/18	RxnMolDQN	Greedy	17.9±6.0
17/18	RxnMolDQN	Random Walk	17.9±3.7
19/20	RxnMolDQN	Epsilon Greedy	19.1±4.0
19/20	RxnMolDQN	Greedy	18.5±5.6
19/20	RxnMolDQN	Random Walk	18.4±3.9
21/22	RxnMolDQN	Epsilon Greedy	18.5±3.9
21/22	RxnMolDQN	Greedy	17.4±5.7
21/22	RxnMolDQN	Random Walk	17.9±3.8
23/24	RxnMolDQN	Epsilon Greedy	18.6±4.0
23/24	RxnMolDQN	Greedy	18.3±6.5
23/24	RxnMolDQN	Random Walk	17.6±3.7

Appendix M: Tanimoto Similarity

Table M1: Tanimoto similarity for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed between initial 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode using the reward function, QED. The values are approximated to three decimal places.

Method Number	Method	Baseline/DQN Methods	Top 1 Average Tanimoto Similarity	Top 10 Average Tanimoto Similarity	Top 20 Average Tanimoto Similarity
5	MolDQN	Naive	0.251±0.159	0.206±0.115	0.195±0.107
6	MolDQN	Bootstrap	0.278±0.165	0.227±0.125	0.211±0.115
7	MolDQN	Naive	0.238±0.130	0.204±0.110	0.192±0.103
8	MolDQN	Bootstrap	0.264±0.145	0.221±0.117	0.206±0.109
5/6	MolDQN	Epsilon Greedy	0.180±0.065	0.160±0.055	0.153±0.052
5/6	MolDQN	Greedy	0.126±0.101	0.126±0.101	0.126±0.101
5/6	MolDQN	Random Walk	0.179±0.062	0.159±0.055	0.152±0.052
7/8	MolDQN	Epsilon Greedy	0.182±0.063	0.161±0.056	0.154±0.053
7/8	MolDQN	Greedy	0.129±0.100	0.129±0.100	0.129±0.100
7/8	MolDQN	Random Walk	0.181±0.063	0.161±0.055	0.154±0.052
9	RxnDQN	Naive	0.578±0.330	0.527±0.360	0.511±0.371
10	RxnDQN	Bootstrap	0.587±0.322	0.526±0.361	0.510±0.371
11	RxnDQN	Naive	0.598±0.329	0.529±0.360	0.513±0.371
12	RxnDQN	Bootstrap	0.602±0.326	0.537±0.355	0.519±0.367
13	RxnDQN	Naive	0.578±0.330	0.527±0.360	0.512±0.371
14	RxnDQN	Bootstrap	0.599±0.321	0.535±0.356	0.518±0.367
15	RxnDQN	Naive	0.585±0.325	0.529±0.359	0.513±0.370
16	RxnDQN	Bootstrap	0.591±0.326	0.533±0.357	0.515±0.368
9/10	RxnDQN	Epsilon Greedy	0.601±0.324	0.537±0.356	0.519±0.367
9/10	RxnDQN	Greedy	0.490±0.369	0.471±0.377	0.457±0.377
9/10	RxnDQN	Random Walk	0.586±0.325	0.534±0.355	0.518±0.366
11/12	RxnDQN	Epsilon Greedy	0.585±0.331	0.535±0.356	0.519±0.366
11/12	RxnDQN	Greedy	0.493±0.368	0.472±0.377	0.458±0.376
11/12	RxnDQN	Random Walk	0.598±0.320	0.535±0.356	0.517±0.367
13/14	RxnDQN	Epsilon Greedy	0.596±0.322	0.531±0.358	0.516±0.368
13/14	RxnDQN	Greedy	0.498±0.366	0.471±0.375	0.460±0.377
13/14	RxnDQN	Random Walk	0.602±0.326	0.534±0.358	0.517±0.368
15/16	RxnDQN	Epsilon Greedy	0.603±0.323	0.535±0.356	0.518±0.367
15/16	RxnDQN	Greedy	0.496±0.366	0.474±0.376	0.461±0.260
15/16	RxnDQN	Random Walk	0.599±0.327	0.531±0.359	0.515±0.234
17	RxnMolDQN	Naive	0.586±0.321	0.457±0.290	0.405±0.260
18	RxnMolDQN	Bootstrap	0.597±0.302	0.425±0.260	0.368±0.237
19	RxnMolDQN	Naive	0.573±0.321	0.460±0.295	0.405±0.260
20	RxnMolDQN	Bootstrap	0.590±0.313	0.434±0.269	0.371±0.231
21	RxnMolDQN	Naive	0.597±0.320	0.464±0.292	0.410±0.256
22	RxnMolDQN	Bootstrap	0.604±0.304	0.426±0.261	0.367±0.236
23	RxnMolDQN	Naive	0.580±0.314	0.456±0.282	0.407±0.122
24	RxnMolDQN	Bootstrap	0.607±0.313	0.436±0.269	0.373±0.175
17/18	RxnMolDQN	Epsilon Greedy	0.485±0.233	0.335±0.141	0.298±0.123
17/18	RxnMolDQN	Greedy	0.421±0.205	0.345±0.180	0.320±0.138
17/18	RxnMolDQN	Random Walk	0.486±0.246	0.333±0.144	0.296±0.170

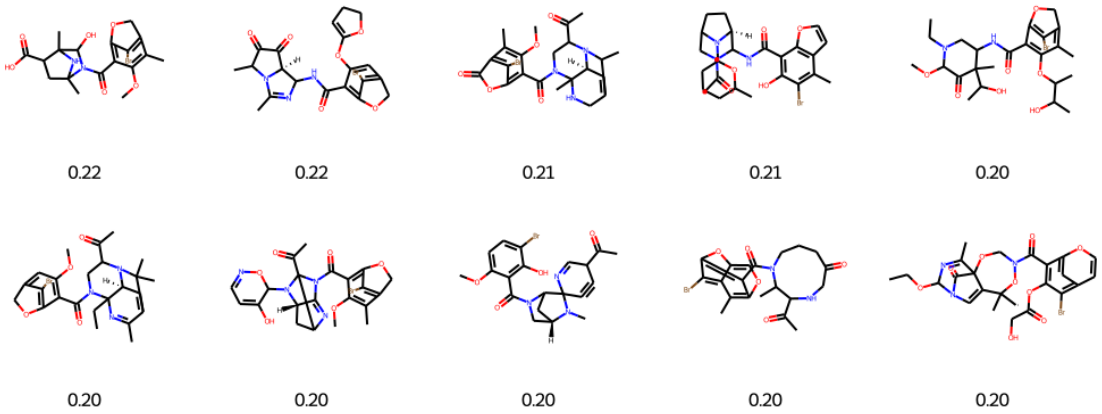
19/20	RxnMolDQN	Epsilon Greedy	0.530±0.262	0.344±0.165	0.302±0.128
19/20	RxnMolDQN	Greedy	0.427±0.193	0.346±0.180	0.317±0.131
19/20	RxnMolDQN	Random Walk	0.506±0.246	0.341±0.151	0.301±0.195
21/22	RxnMolDQN	Epsilon Greedy	0.515±0.255	0.342±0.155	0.301±0.136
21/22	RxnMolDQN	Greedy	0.437±0.207	0.371±0.197	0.345±0.130
21/22	RxnMolDQN	Random Walk	0.526±0.250	0.347±0.161	0.305±0.180
23/24	RxnMolDQN	Epsilon Greedy	0.514±0.253	0.343±0.155	0.302±0.130
23/24	RxnMolDQN	Greedy	0.421±0.191	0.350±0.182	0.327±0.260
23/24	RxnMolDQN	Random Walk	0.512±0.250	0.343±0.153	0.301±0.234

Table M2: Tanimoto similarity for MolDQN (methods 5-8), RxnDQN (methods 9-16), and RxnMolDQN (methods 17-24) together with its corresponding baselines (epsilon greedy, greedy, and random walk) computed between initial 100 chemically diverse compounds from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode using the reward function, penalized logP. The values are approximated to three decimal places.

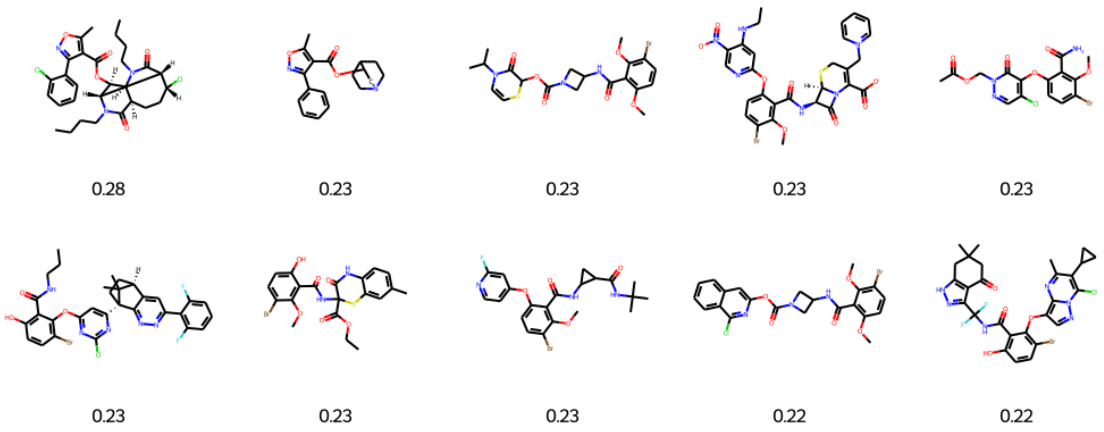
Method Number	Method	Baseline/DQN Methods	Top 1 Average Tanimoto Similarity	Top 10 Average Tanimoto Similarity	Top 20 Average Tanimoto Similarity
5	MolDQN	Naive	0.251±0.159	0.206±0.115	0.195±0.107
6	MolDQN	Bootstrap	0.278±0.165	0.227±0.125	0.211±0.115
7	MolDQN	Naive	0.238±0.130	0.204±0.110	0.192±0.103
8	MolDQN	Bootstrap	0.264±0.145	0.221±0.117	0.206±0.109
5/6	MolDQN	Epsilon Greedy	0.180±0.065	0.160±0.055	0.153±0.052
5/6	MolDQN	Greedy	0.126±0.101	0.126±0.101	0.126±0.101
5/6	MolDQN	Random Walk	0.179±0.062	0.159±0.055	0.152±0.052
7/8	MolDQN	Epsilon Greedy	0.182±0.063	0.161±0.056	0.154±0.053
7/8	MolDQN	Greedy	0.129±0.100	0.129±0.100	0.129±0.100
7/8	MolDQN	Random Walk	0.181±0.063	0.161±0.055	0.154±0.052
9	RxnDQN	Naive	0.578±0.330	0.527±0.360	0.511±0.371
10	RxnDQN	Bootstrap	0.587±0.322	0.526±0.361	0.510±0.371
11	RxnDQN	Naive	0.598±0.329	0.529±0.360	0.513±0.371
12	RxnDQN	Bootstrap	0.602±0.326	0.537±0.355	0.519±0.367
13	RxnDQN	Naive	0.578±0.330	0.527±0.360	0.512±0.371
14	RxnDQN	Bootstrap	0.599±0.321	0.535±0.356	0.518±0.367
15	RxnDQN	Naive	0.585±0.325	0.529±0.359	0.513±0.370
16	RxnDQN	Bootstrap	0.591±0.326	0.533±0.357	0.515±0.368
9/10	RxnDQN	Epsilon Greedy	0.601±0.324	0.537±0.356	0.519±0.367
9/10	RxnDQN	Greedy	0.490±0.369	0.471±0.377	0.457±0.377
9/10	RxnDQN	Random Walk	0.586±0.325	0.534±0.355	0.518±0.366
11/12	RxnDQN	Epsilon Greedy	0.585±0.331	0.535±0.356	0.519±0.366
11/12	RxnDQN	Greedy	0.493±0.368	0.472±0.377	0.458±0.376
11/12	RxnDQN	Random Walk	0.598±0.320	0.535±0.356	0.517±0.367
13/14	RxnDQN	Epsilon Greedy	0.596±0.322	0.531±0.358	0.516±0.368
13/14	RxnDQN	Greedy	0.498±0.366	0.471±0.375	0.460±0.377
13/14	RxnDQN	Random Walk	0.602±0.326	0.534±0.358	0.517±0.368
15/16	RxnDQN	Epsilon Greedy	0.603±0.323	0.535±0.356	0.518±0.367
15/16	RxnDQN	Greedy	0.496±0.366	0.474±0.376	0.461±0.377
15/16	RxnDQN	Random Walk	0.599±0.327	0.531±0.359	0.515±0.368
17	RxnMolDQN	Naive	0.586±0.321	0.457±0.290	0.405±0.260
18	RxnMolDQN	Bootstrap	0.597±0.302	0.425±0.260	0.368±0.234
19	RxnMolDQN	Naive	0.573±0.321	0.460±0.295	0.405±0.260
20	RxnMolDQN	Bootstrap	0.590±0.313	0.434±0.269	0.371±0.237
21	RxnMolDQN	Naive	0.597±0.320	0.464±0.292	0.410±0.260
22	RxnMolDQN	Bootstrap	0.604±0.304	0.426±0.261	0.367±0.231
23	RxnMolDQN	Naive	0.580±0.314	0.456±0.282	0.407±0.256
24	RxnMolDQN	Bootstrap	0.607±0.313	0.436±0.269	0.373±0.236
17/18	RxnMolDQN	Epsilon Greedy	0.485±0.233	0.335±0.141	0.298±0.122
17/18	RxnMolDQN	Greedy	0.421±0.205	0.345±0.180	0.320±0.175
17/18	RxnMolDQN	Random Walk	0.486±0.246	0.333±0.144	0.296±0.123
19/20	RxnMolDQN	Epsilon Greedy	0.530±0.262	0.344±0.165	0.302±0.138
19/20	RxnMolDQN	Greedy	0.427±0.193	0.346±0.180	0.317±0.170
19/20	RxnMolDQN	Random Walk	0.506±0.246	0.341±0.151	0.301±0.128

21/22	RxnMolDQN	Epsilon Greedy	0.515±0.255	0.342±0.155	0.301±0.131
21/22	RxnMolDQN	Greedy	0.437±0.207	0.371±0.197	0.345±0.195
21/22	RxnMolDQN	Random Walk	0.526±0.250	0.347±0.161	0.305±0.136
23/24	RxnMolDQN	Epsilon Greedy	0.514±0.253	0.343±0.155	0.302±0.130
23/24	RxnMolDQN	Greedy	0.421±0.191	0.350±0.182	0.327±0.180
23/24	RxnMolDQN	Random Walk	0.512±0.250	0.343±0.153	0.301±0.130

(a)



(b)



(c)

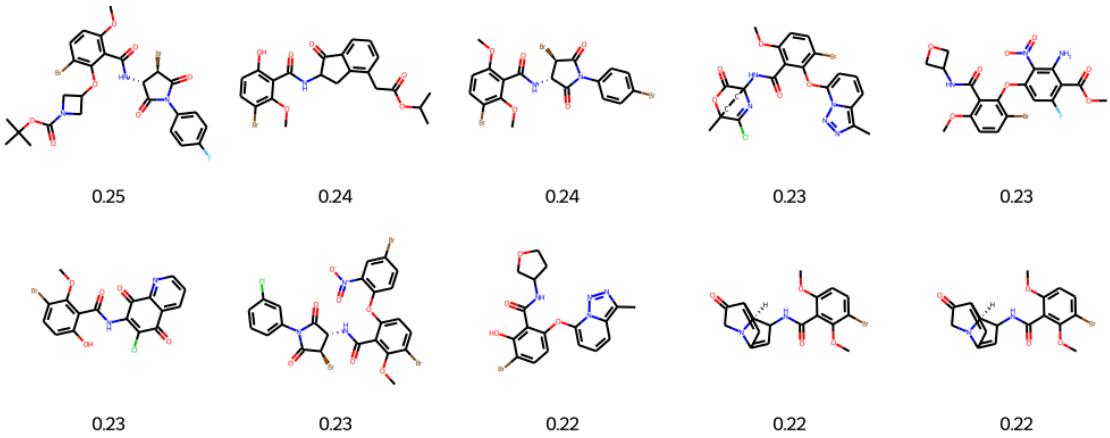
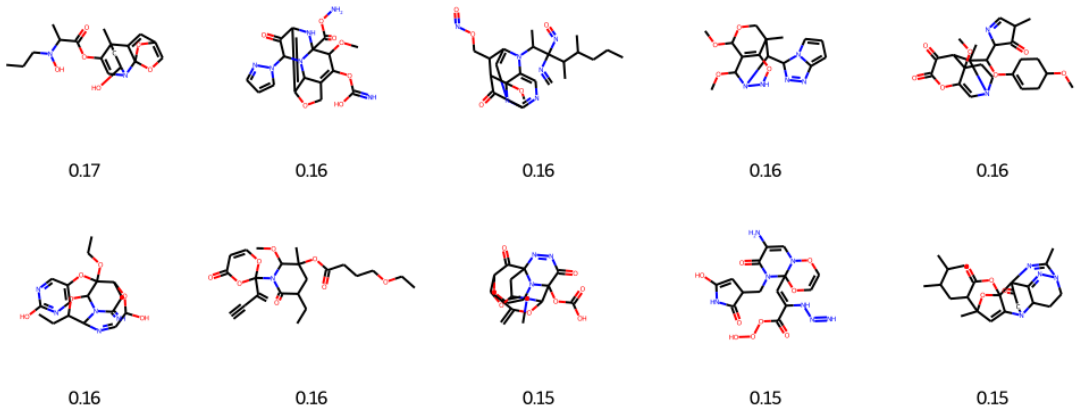
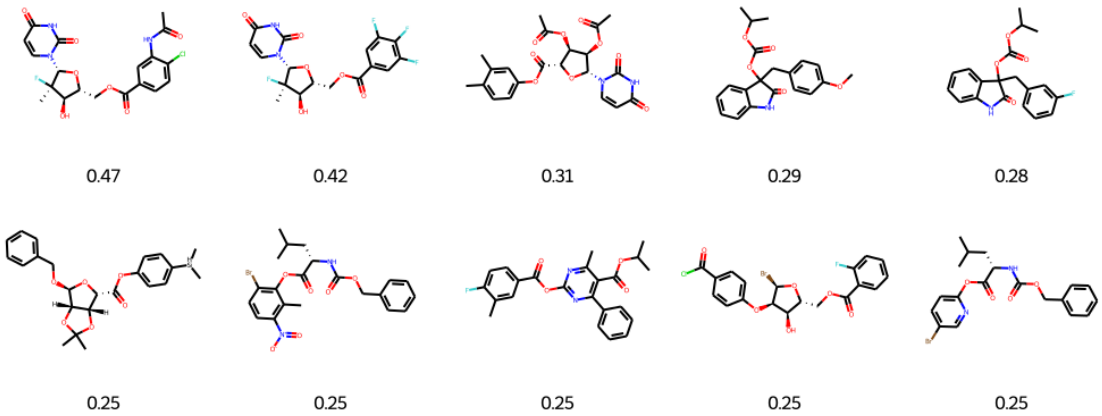


Figure M3: Top 10 Tanimoto similarity computed between flucloxacilin/DB00301 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

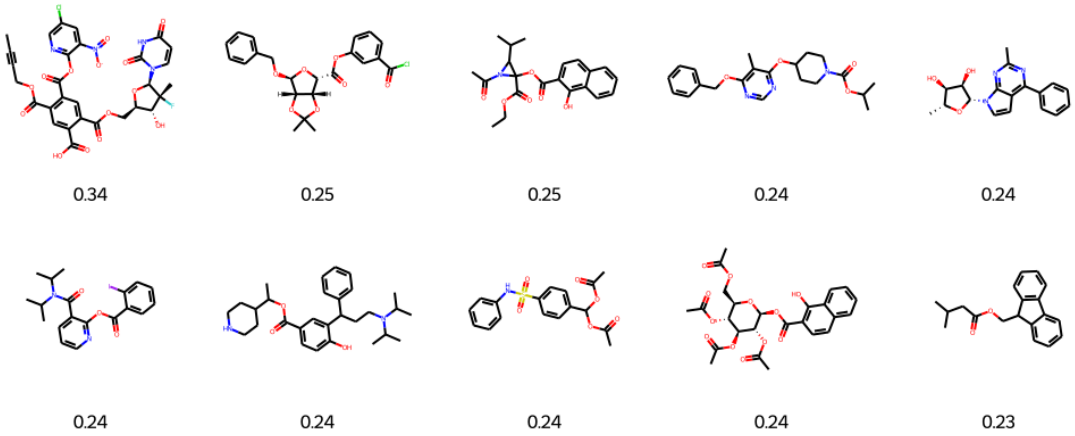
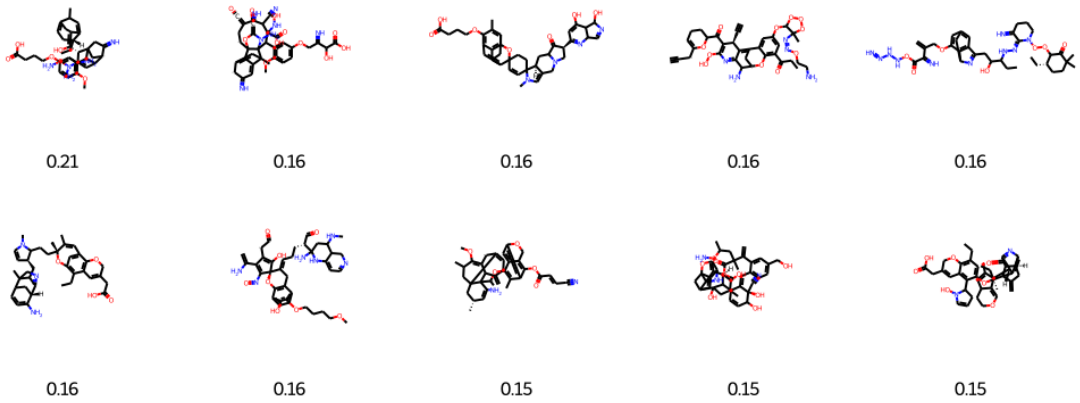
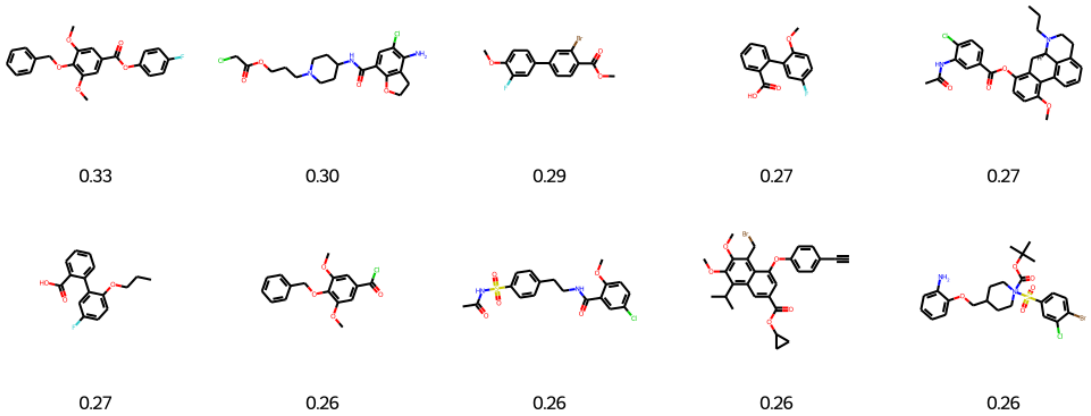


Figure M4: Top 10 Tanimoto similarity computed between sofosbuvir/DB08934 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

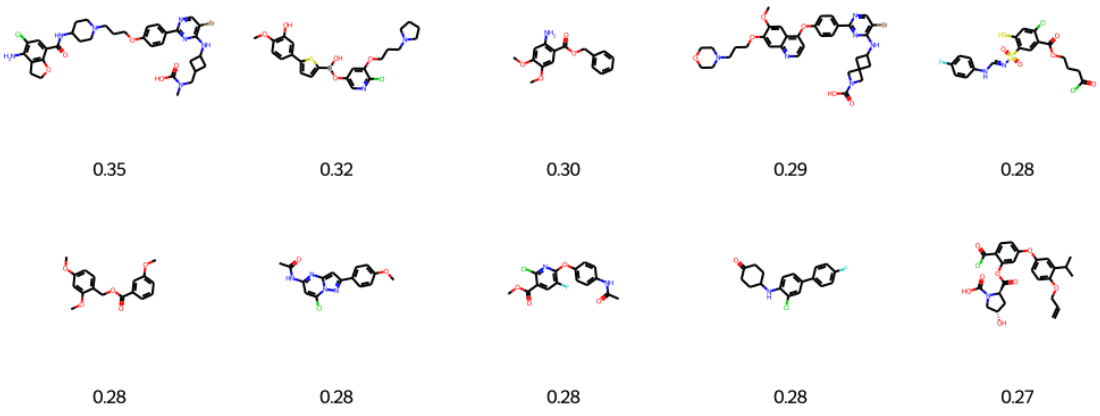


Figure M5: Top 10 Tanimoto similarity computed between cisapride/DB00604 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

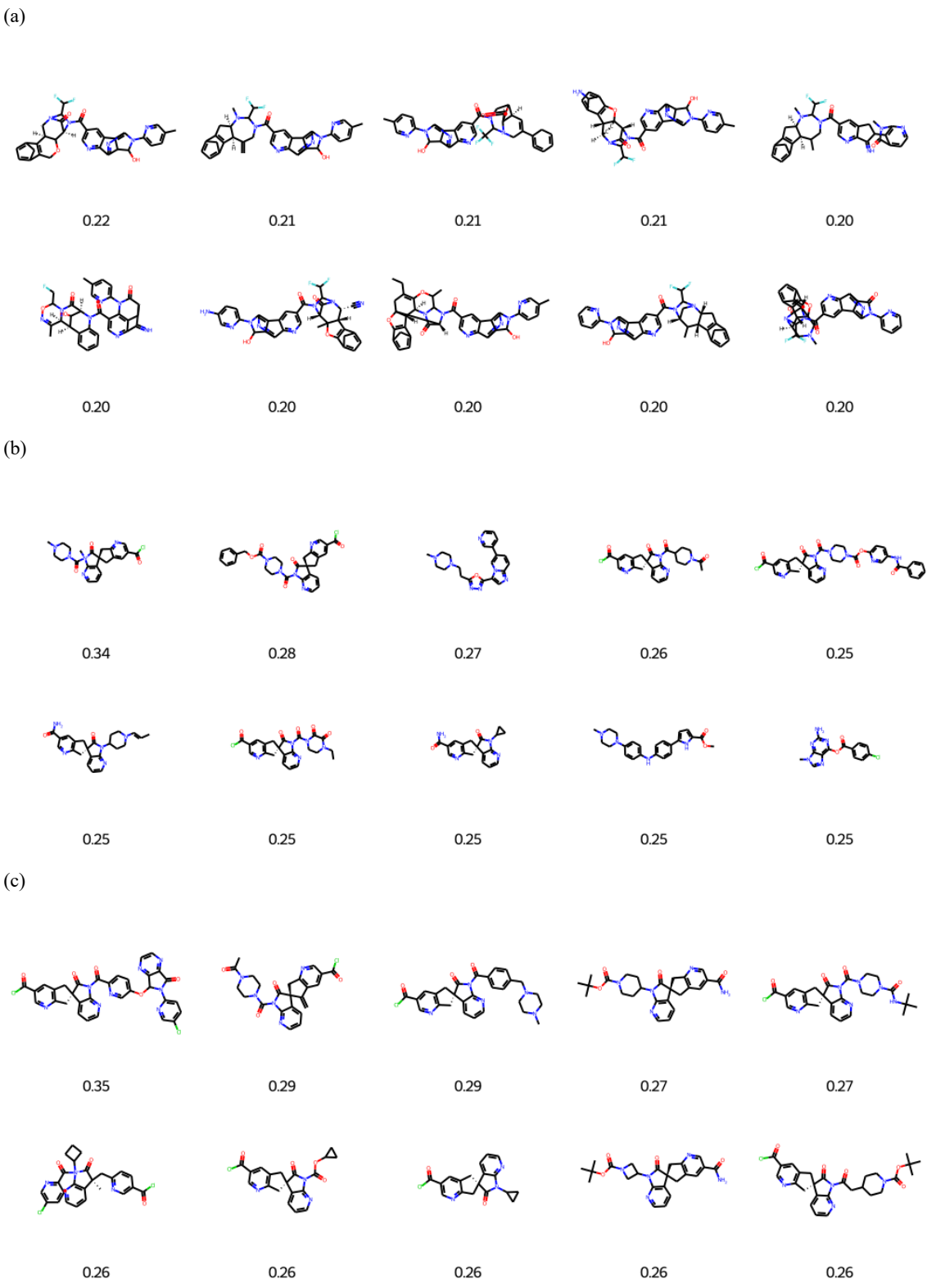
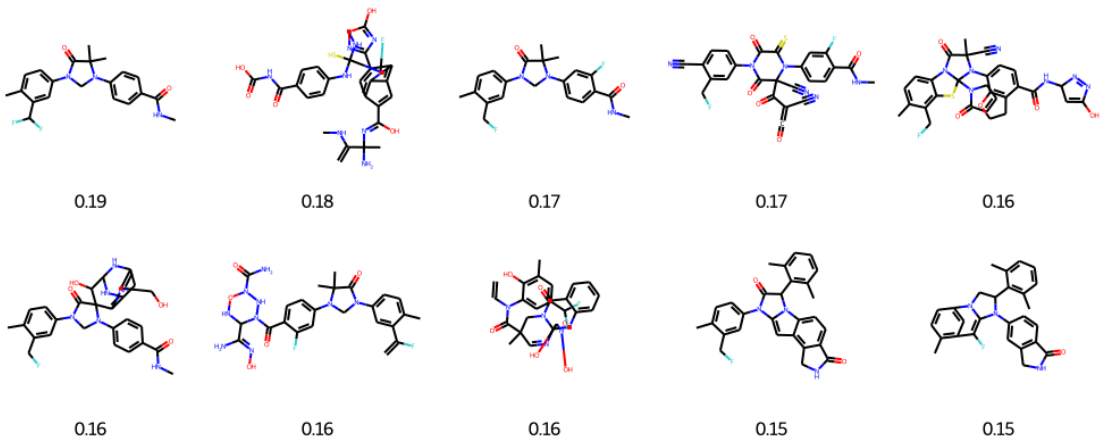
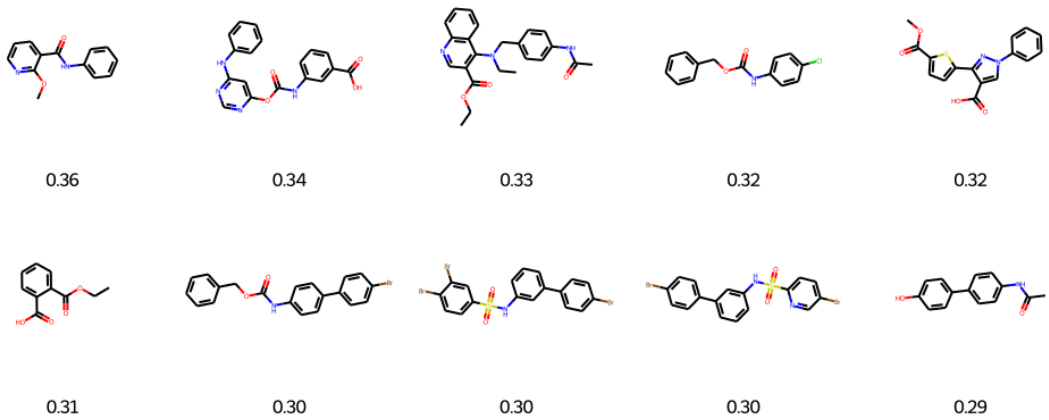


Figure M6: Top 10 Tanimoto similarity computed between zopiclone/DB01198 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

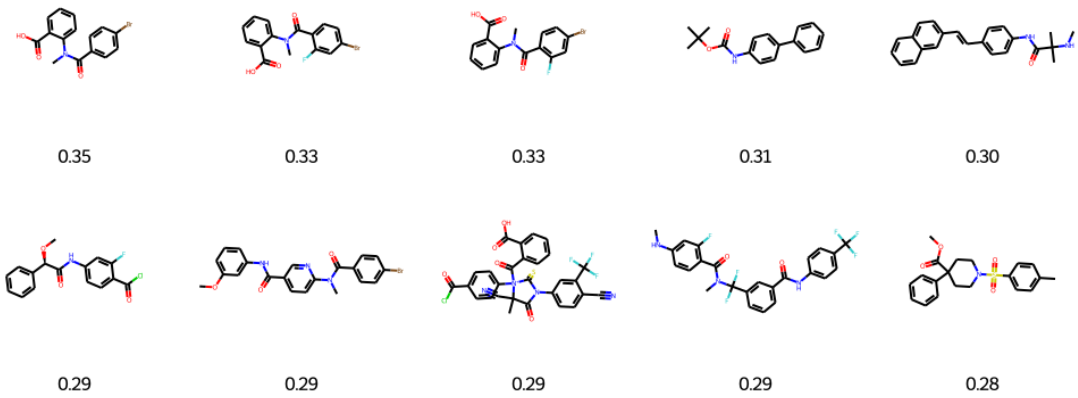
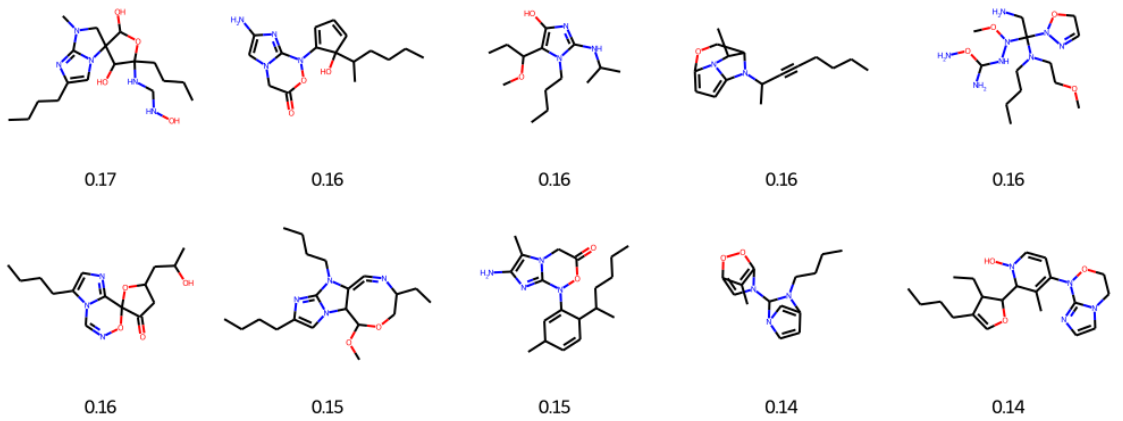
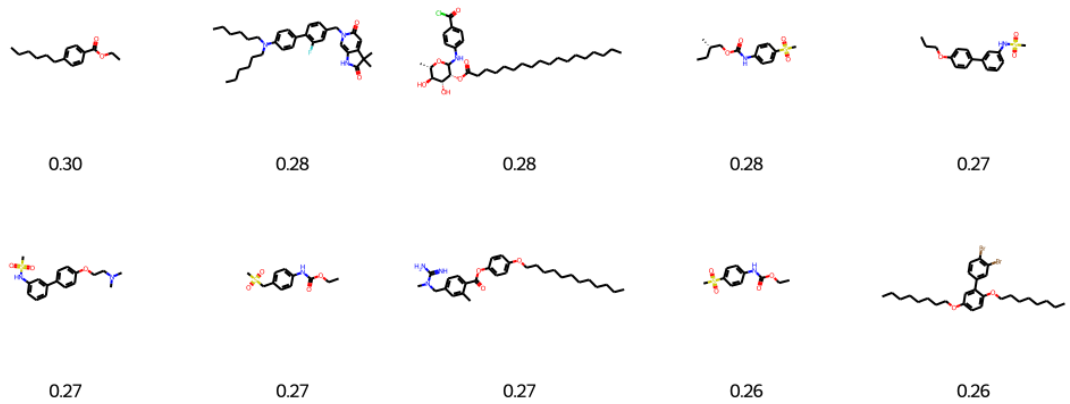


Figure M7: Top 10 Tanimoto similarity computed between phthalylsulfathiazole/DB13248 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

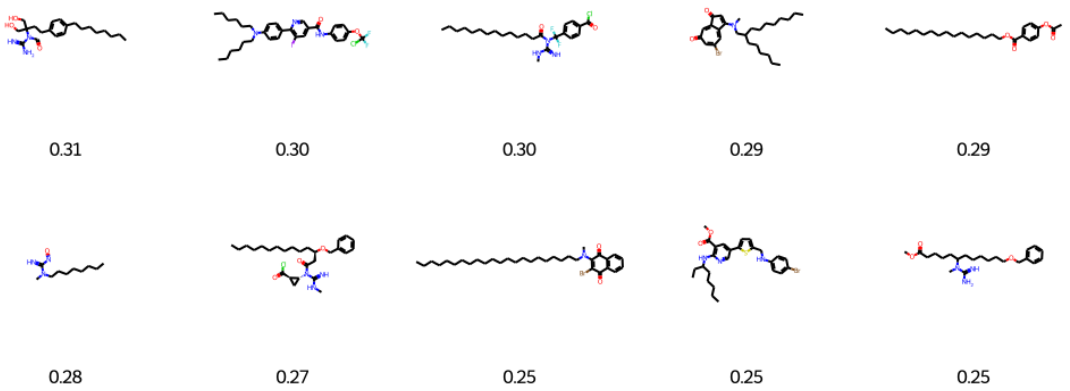
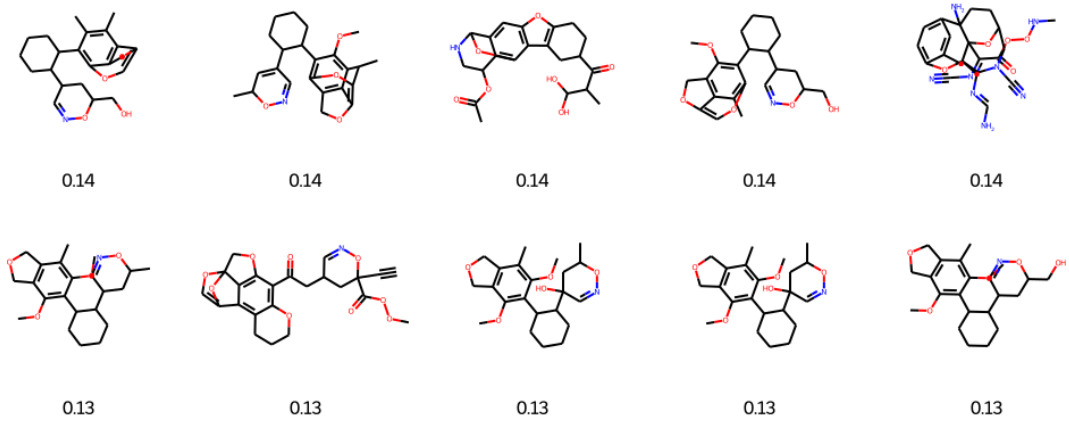
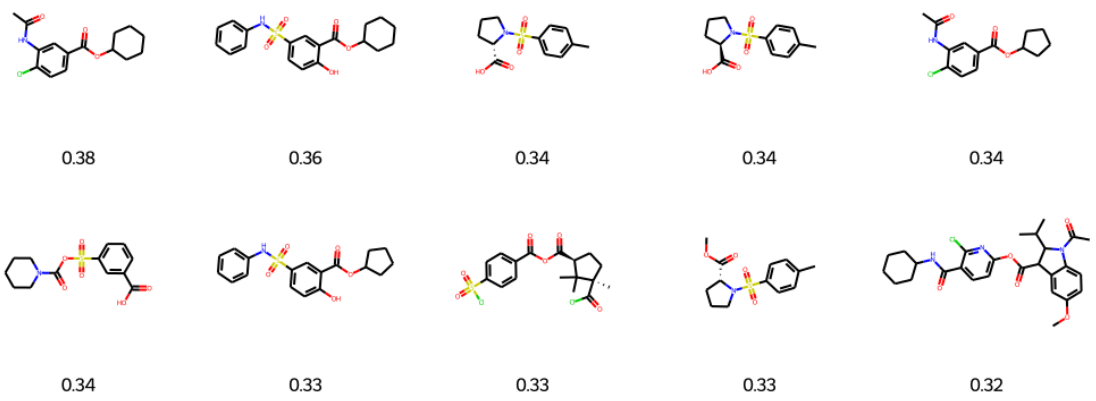


Figure M8: Top 10 Tanimoto similarity computed between ibutilide/DB00308 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

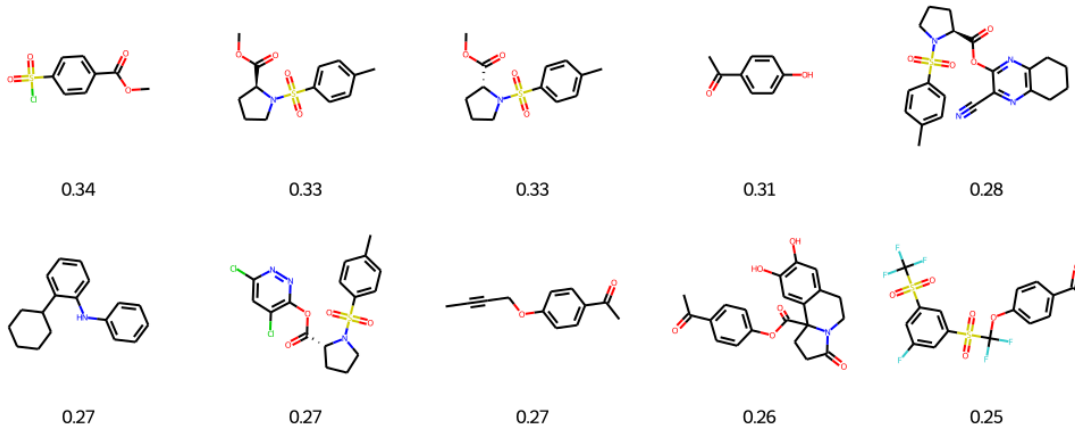


Figure M9: Top 10 Tanimoto similarity computed between acetohexamide/DB00414 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

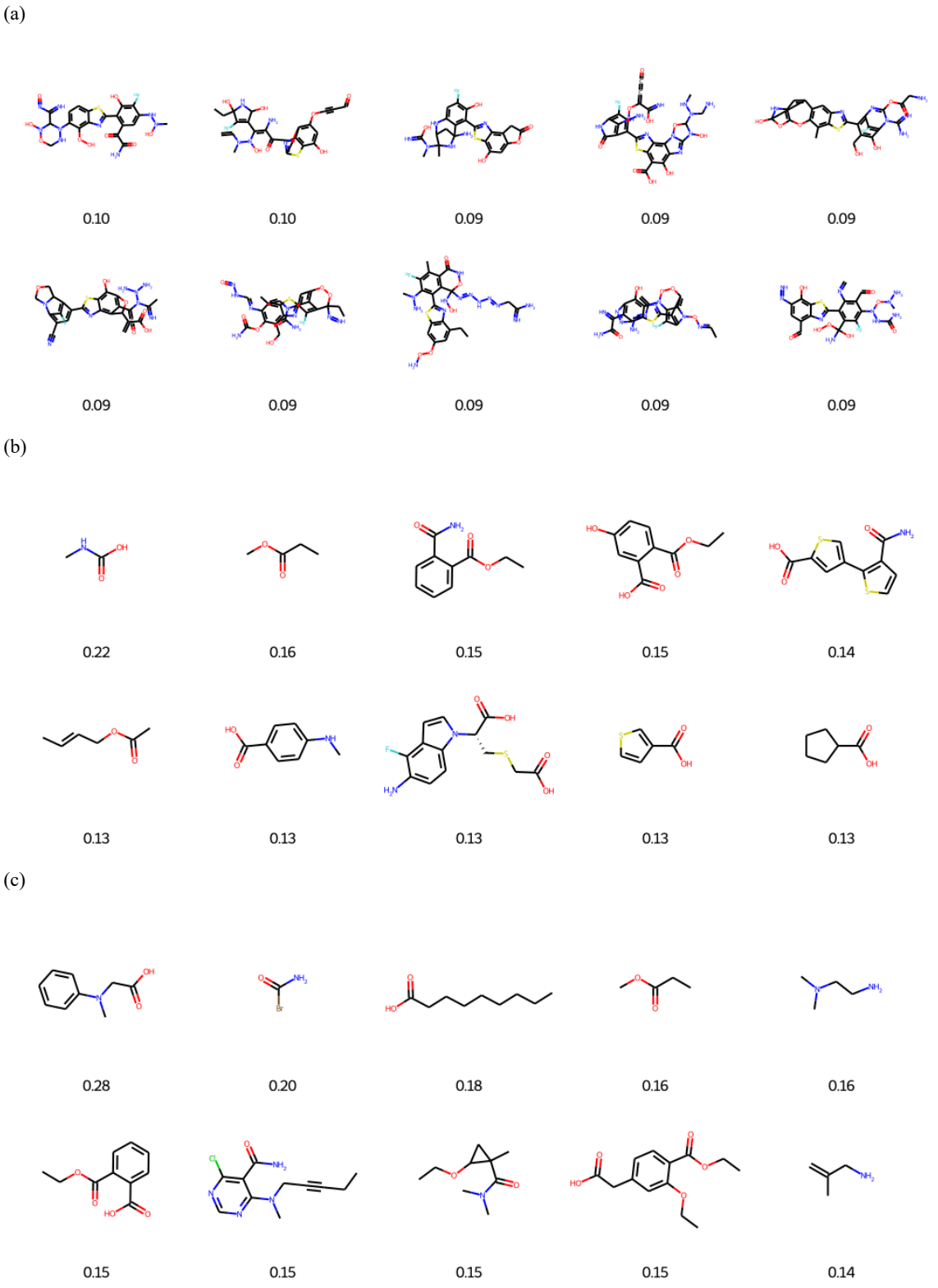
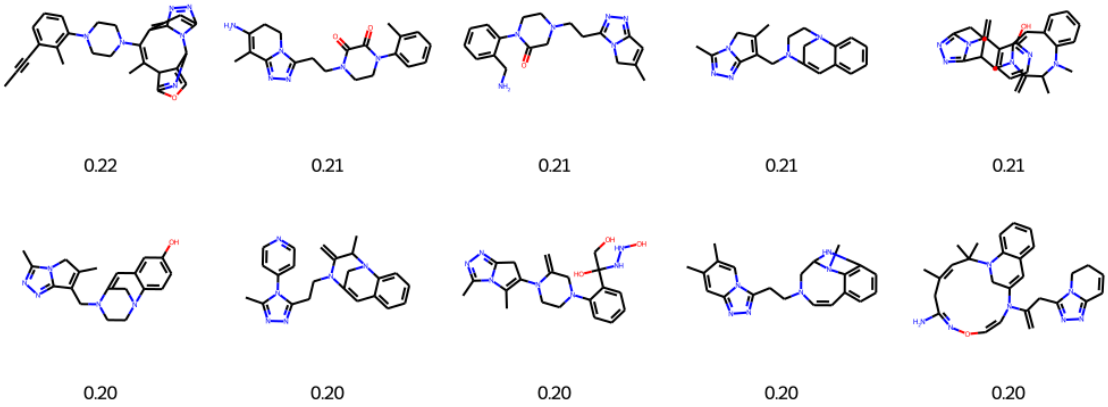
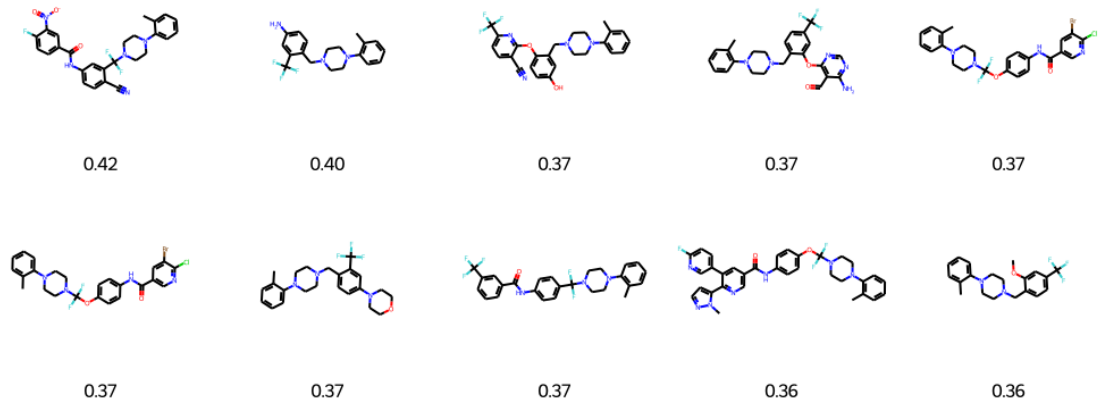


Figure M10: Top 10 Tanimoto similarity computed between creatine/DB00148 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

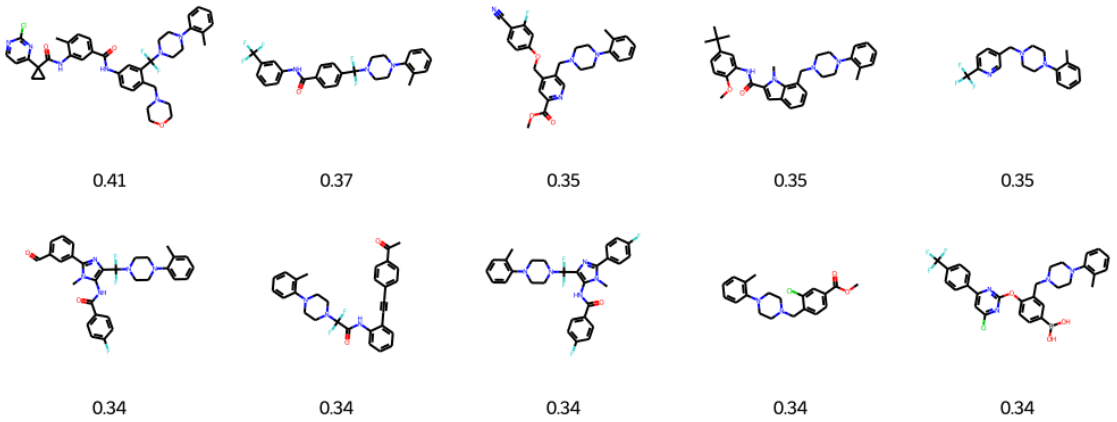
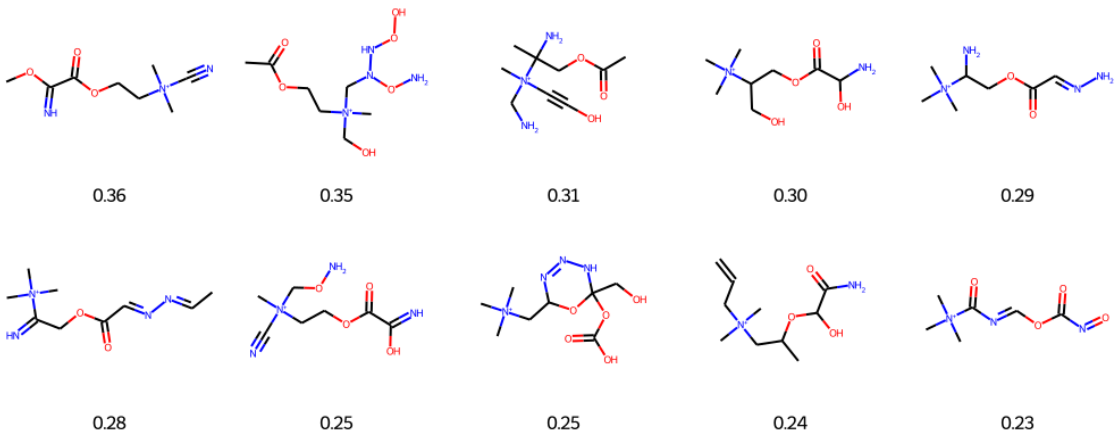
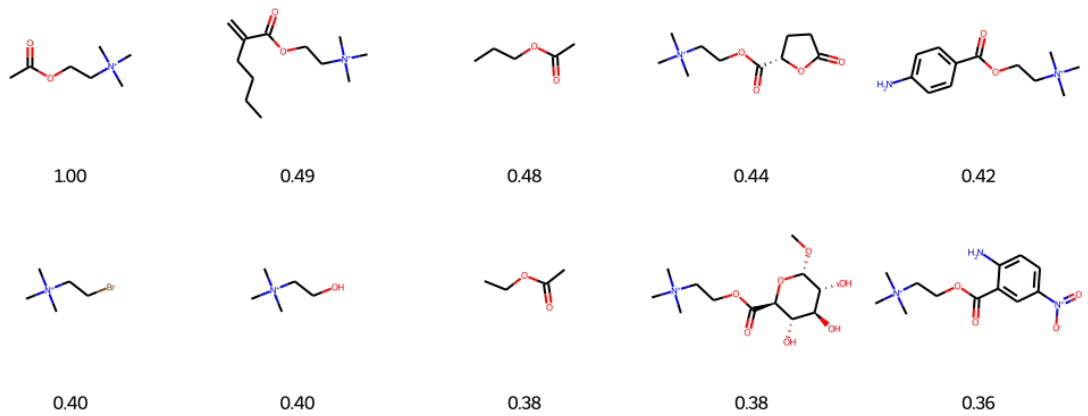


Figure M11: Top 10 Tanimoto similarity computed between ponatinib/DB08901 from DrugBank (Wishart *et al.*, 2018) and generated molecule at each episode where the reward function is QED: (a) MolDQN, (b) RxnDQN, and (c) RxnMolDQN.

(a)



(b)



(c)

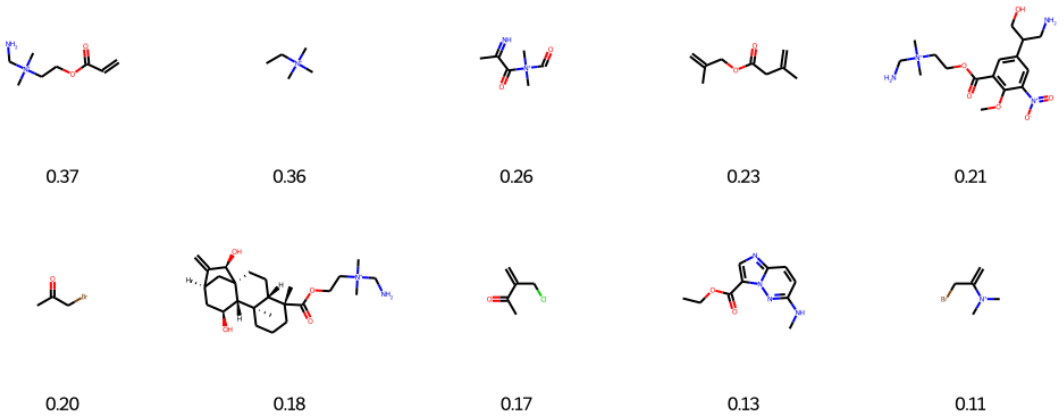


Figure M12: Top 10 Tanimoto similarity computed between ACH/2ha4 from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule at each episode where the reward function is predicted ligand efficiency: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

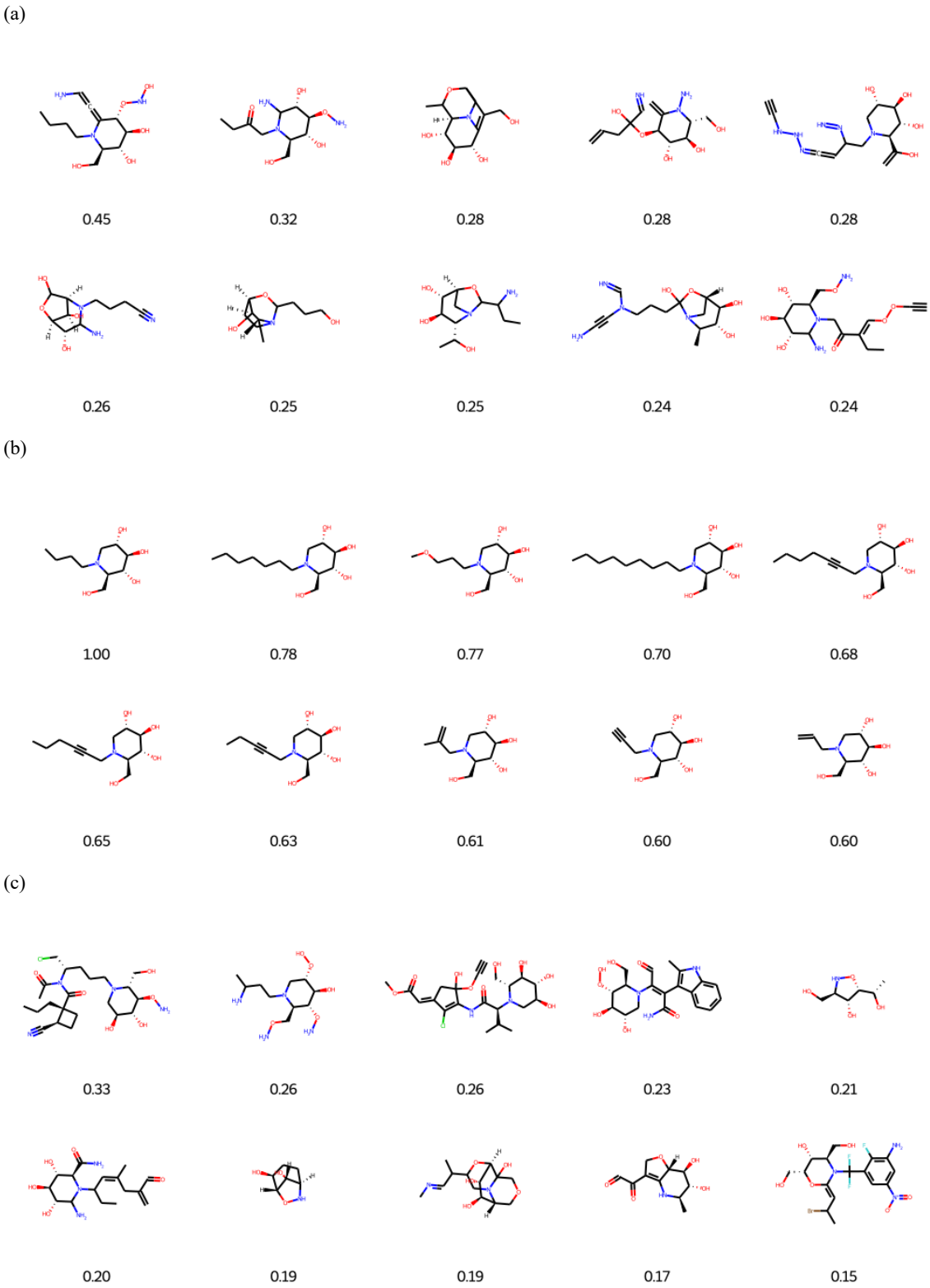
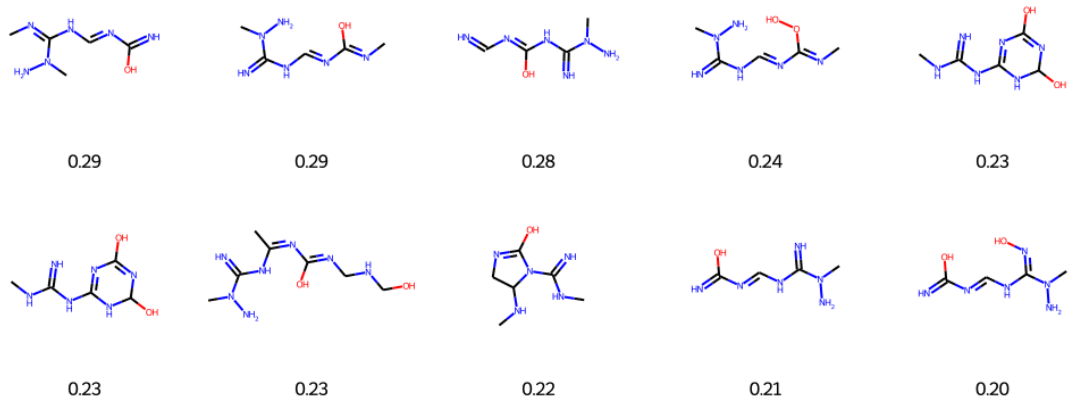
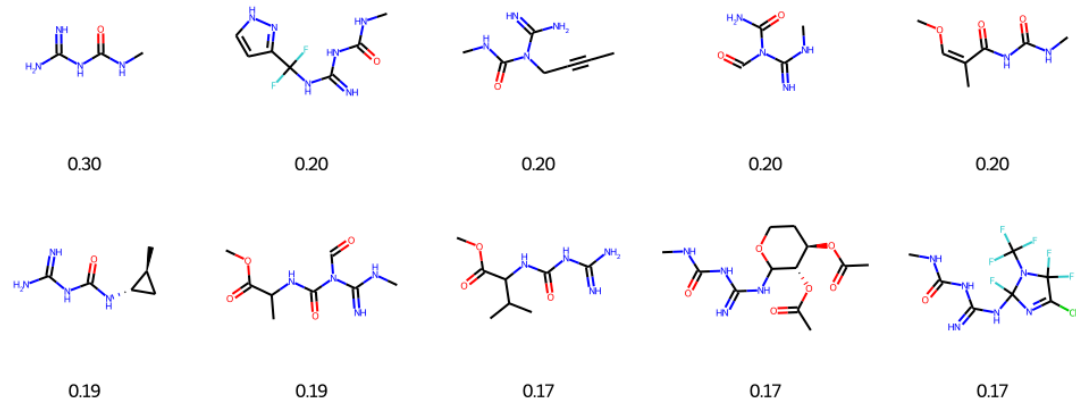


Figure M13: Top 10 Tanimoto similarity computed between NBV/2v3d from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule at each episode where the reward function is predicted ligand efficiency: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

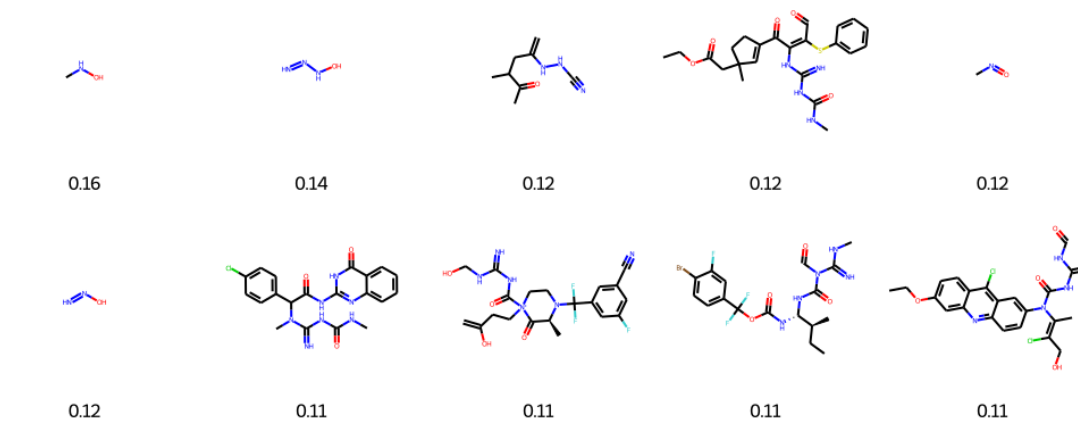
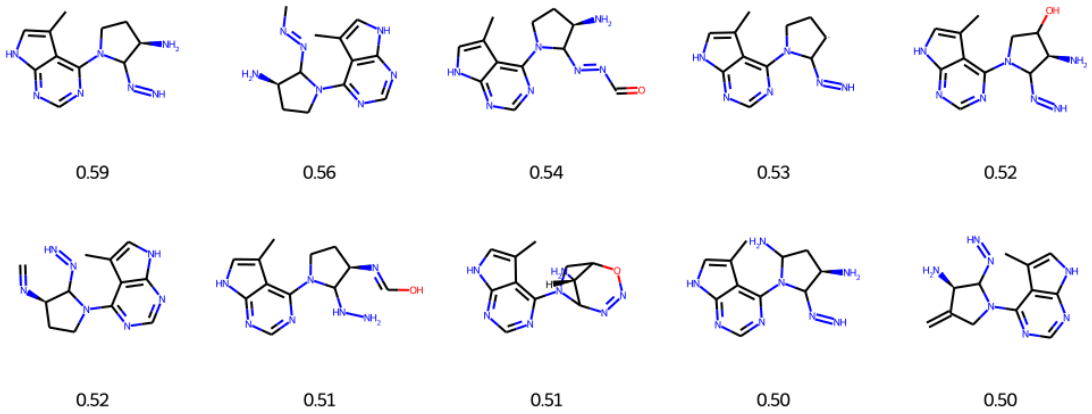
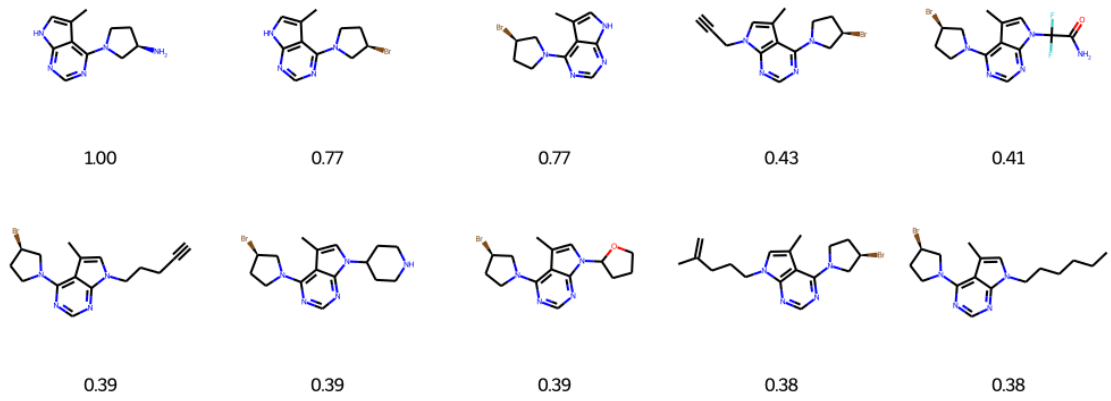


Figure M14: Top 10 Tanimoto similarity computed between XRG/3ch9 from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule at each episode where the reward function is predicted ligand efficiency: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

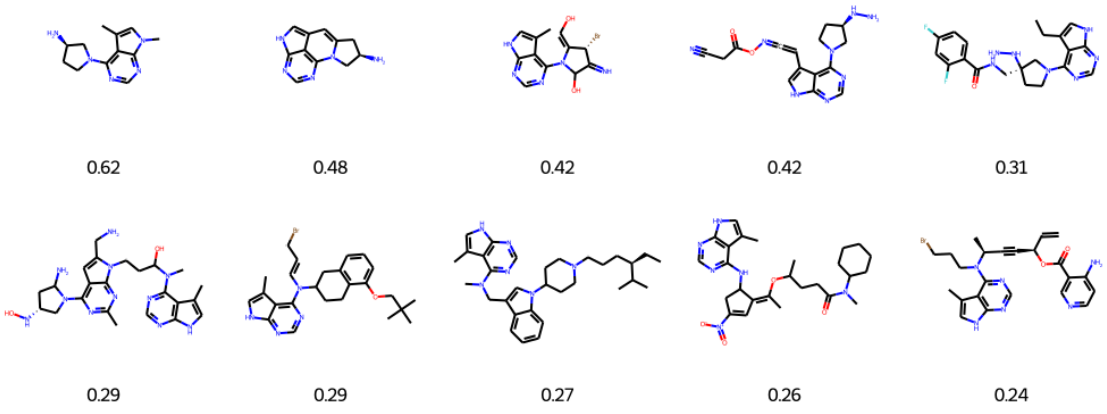
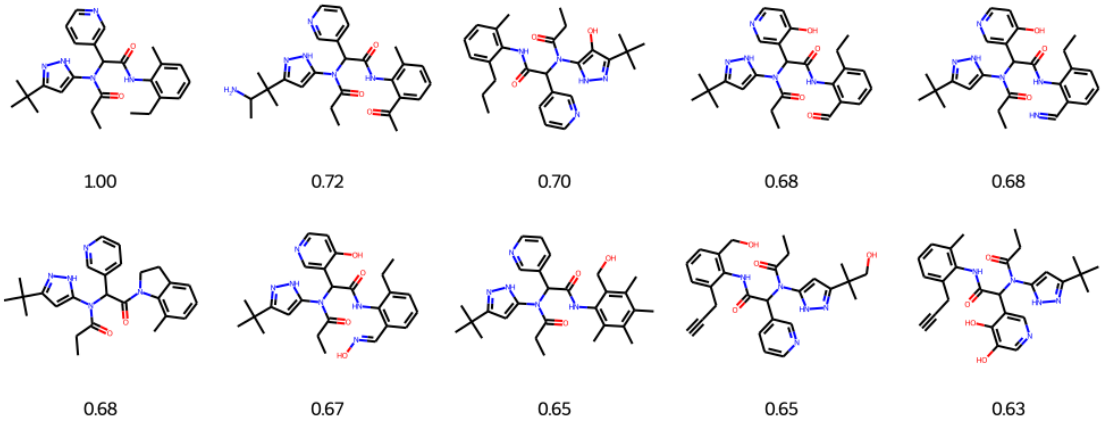
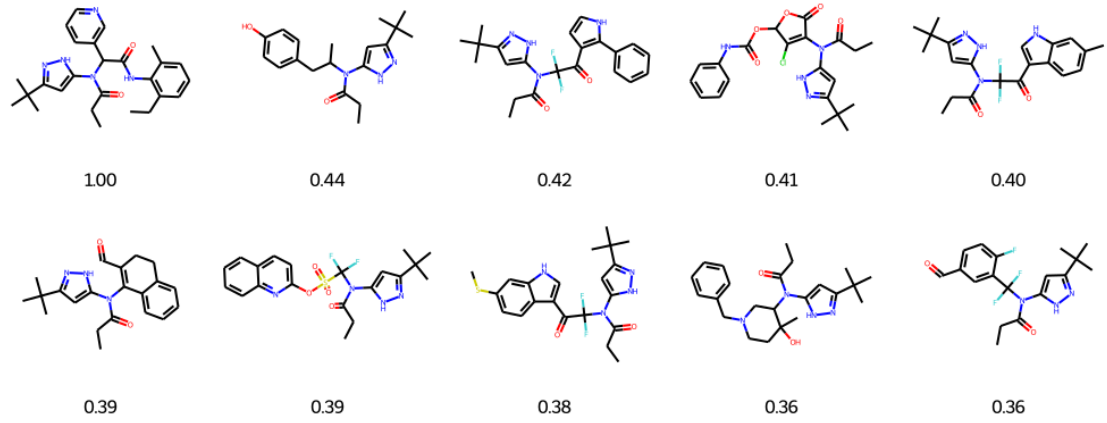


Figure M15: Top 10 Tanimoto similarity computed between XFE/3mv5 from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule at each episode where the reward function is predicted ligand efficiency: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

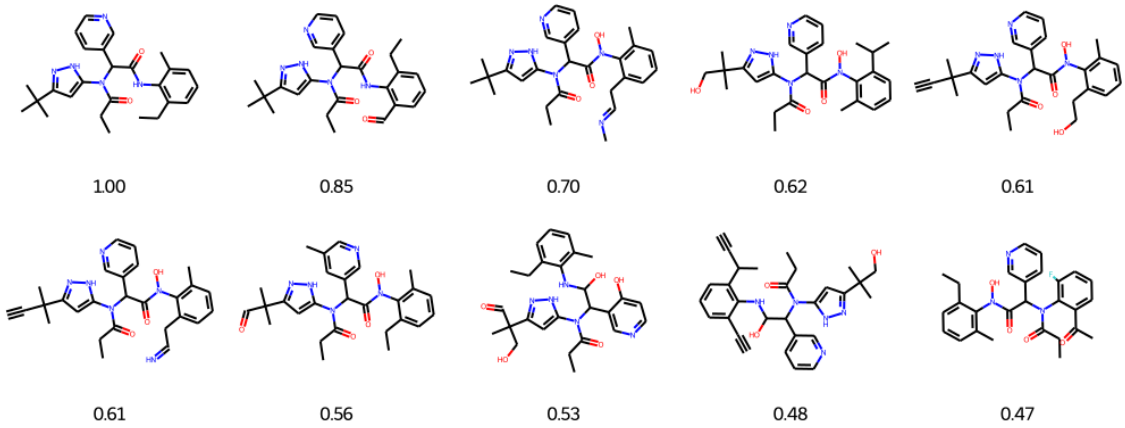


Figure M16: Top 10 Tanimoto similarity computed between Mpro-x2705_0A from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) and generated molecule at each episode where the reward function is predicted ligand efficiency: (a) c3D-MolDQN, (b) c3D-RxnDQN, and (c) c3D-RxnMolDQN.

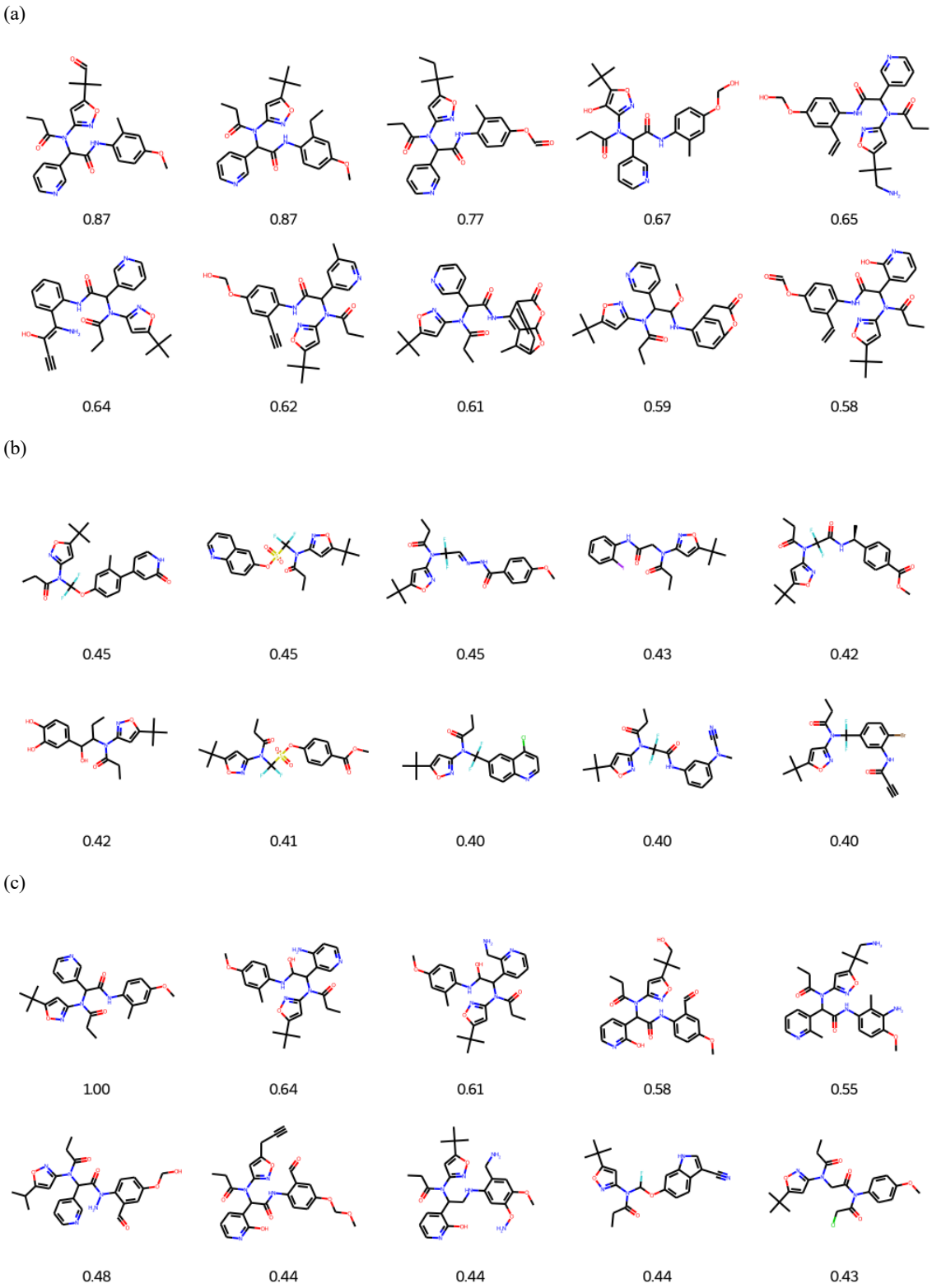
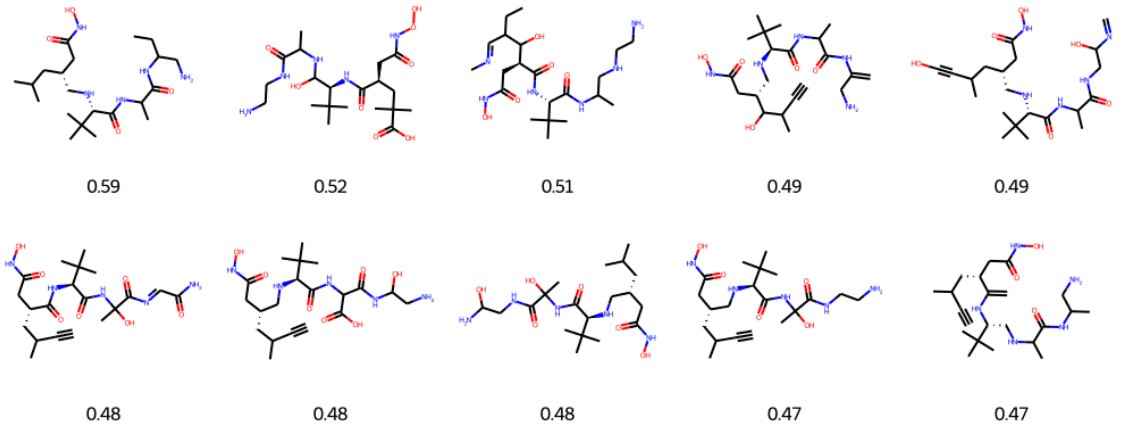
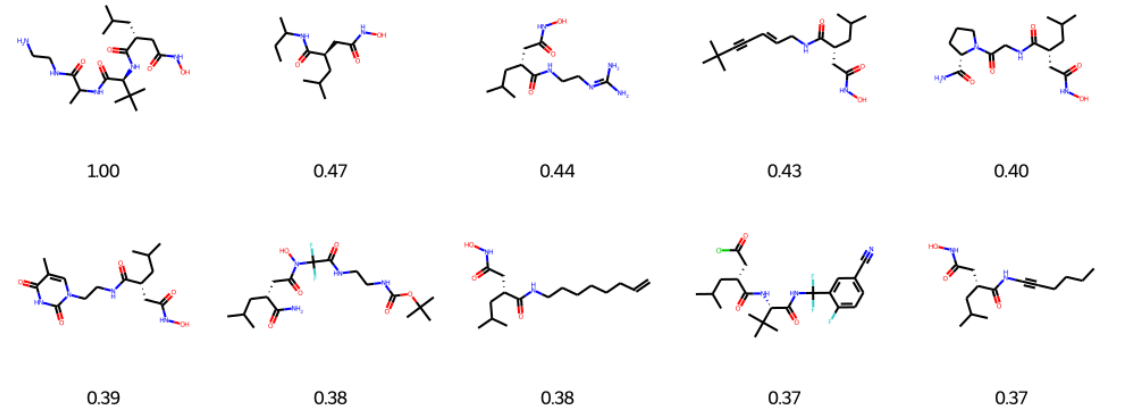


Figure M17: Top 10 Tanimoto similarity computed between Mpro-x2694_0A from SARS-CoV-2 main protease (Douangamath *et al.*, 2020) and generated molecule at each episode where the reward function is predicted ligand efficiency: (a) c3D-MolDQN, (b) c3D-RxnDQN, and (c) c3D-RxnMolDQN.

(a)



(b)



(c)

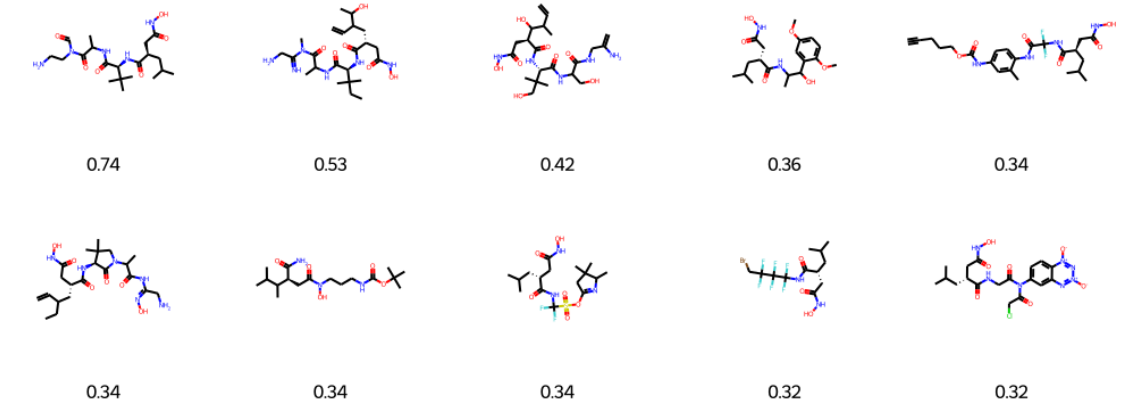
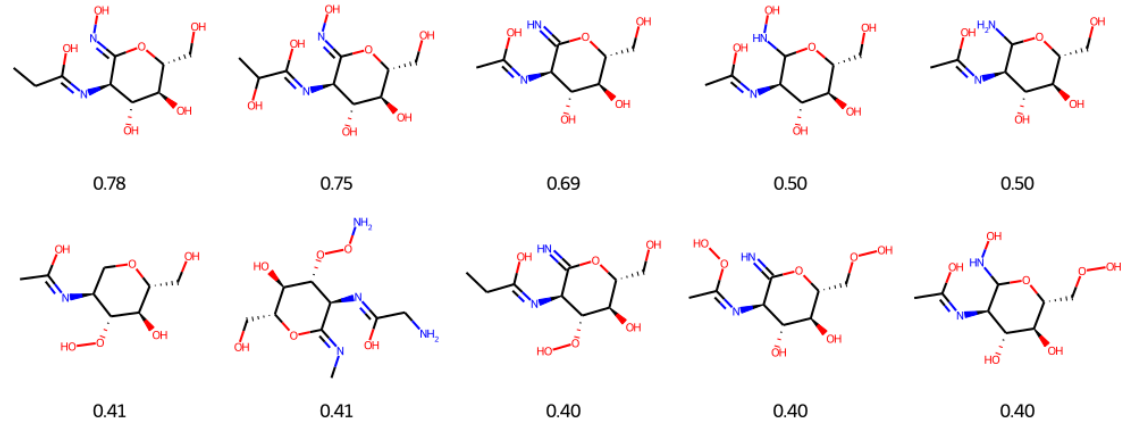
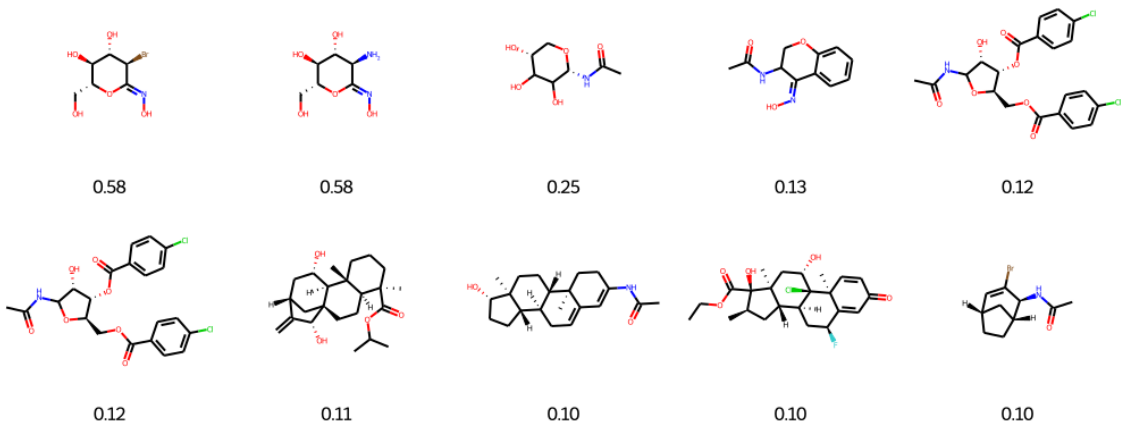


Figure M18: Top 10 Tanimoto similarity computed between INN/3kme from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule at each episode where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

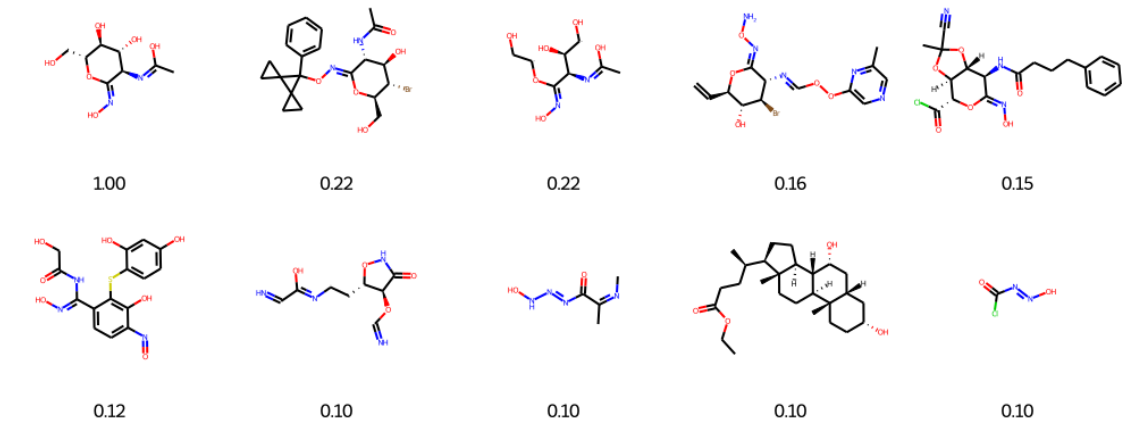
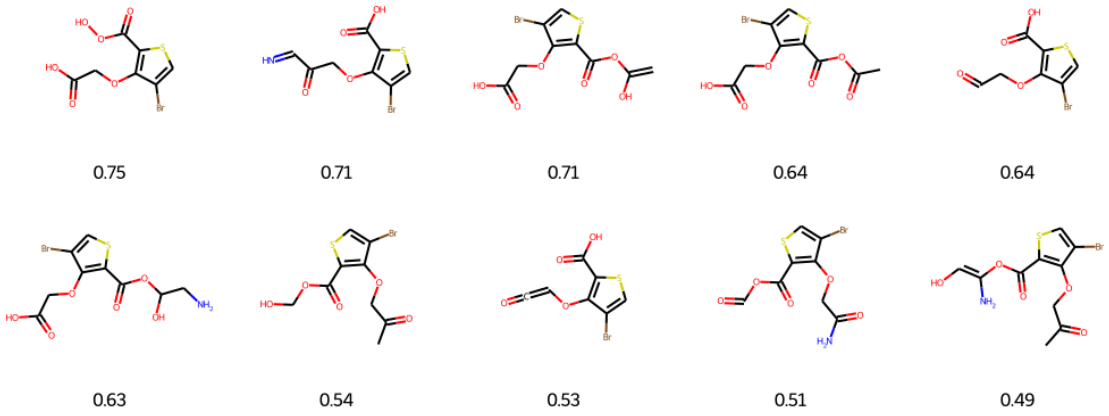
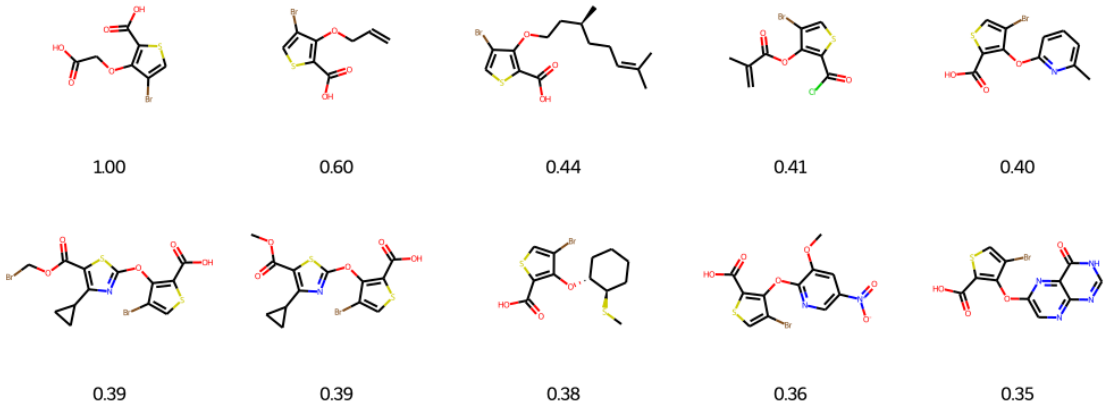


Figure M19: Top 10 Tanimoto similarity computed between LOG/2xm2 from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule at each episode where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

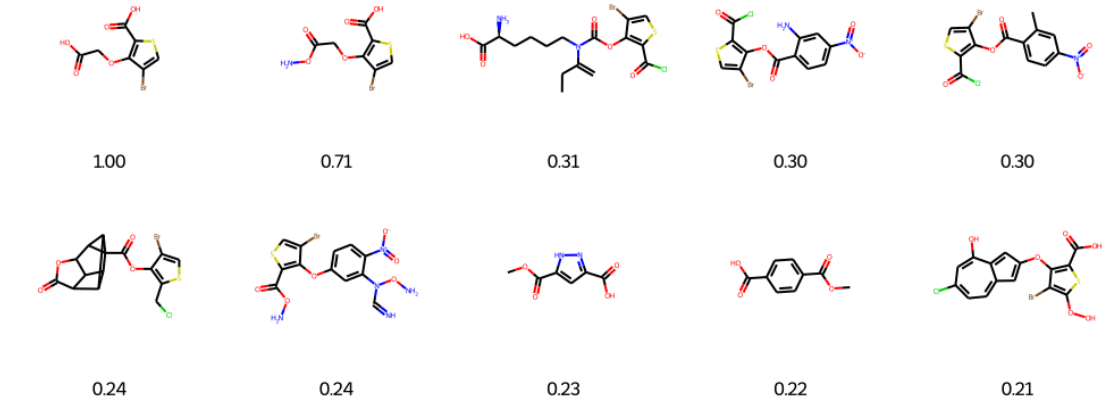
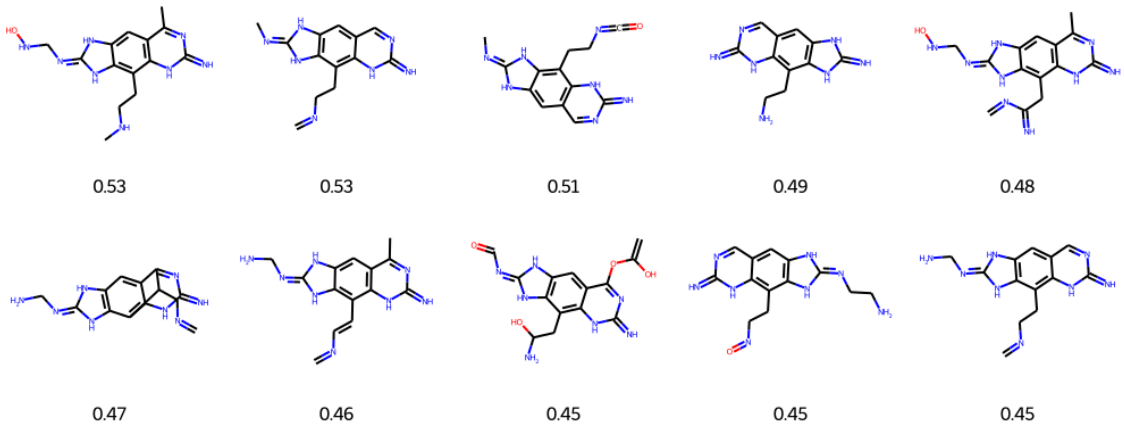
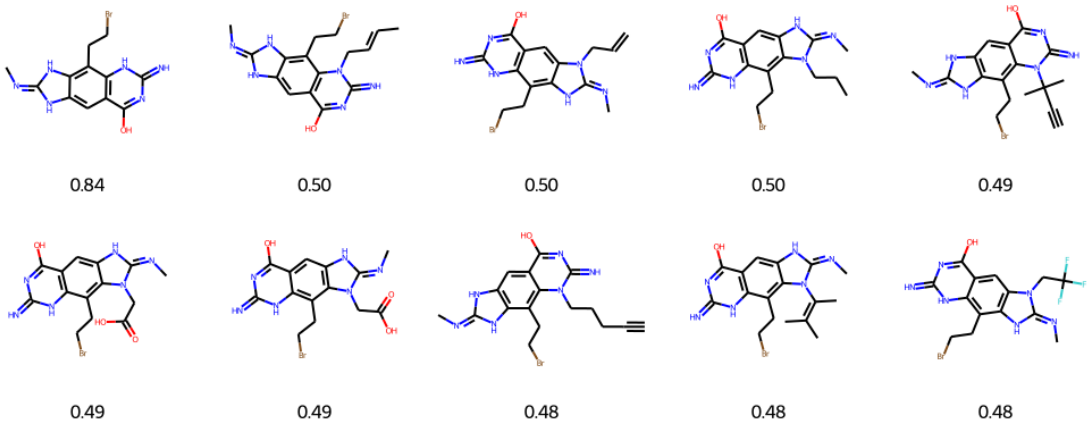


Figure M20: Top 10 Tanimoto similarity computed between 512/2hb1 from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule at each episode where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

(a)



(b)



(c)

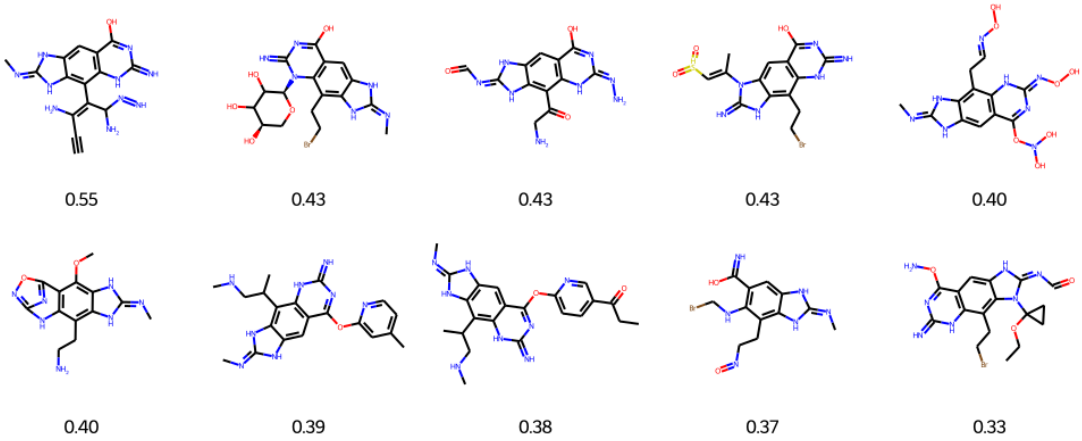


Figure M21: Top 10 Tanimoto similarity computed between 2MQ/3gc5 from the Malhotra set (Malhotra and Karanicolas, 2017) and generated molecule at each episode where the reward function is normalized SuCOS: (a) nc3D-MolDQN, (b) nc3D-RxnDQN, and (c) nc3D-RxnMolDQN.

Appendix N: Sample Workflow for c3D-DQN Methods

Appendix N shows the sample SDF, PDB, and PDBQT files from docking workflow used in c3D-DQN methods.

(a)

```
OpenBabel11162110483D

 75 77  0  0  1  0  0  0  0  0999 V2000
    1.0005   -0.0160    0.0865 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.3597    0.8866    0.6238 O   0  0  0  0  0  0  0  0  0  0  0  0
    2.5200   -0.0677    0.0513 C   0  0  3  0  0  0  0  0  0  0  0  0
    2.9588   -1.2891   -0.6555 N   0  0  0  0  0  0  0  0  0  0  0  0
    3.0694    1.2171   -0.6026 C   0  0  0  0  0  0  0  0  0  0  0  0
    4.8935    1.1895   -0.5145 S   0  0  0  0  0  0  0  0  0  0  0  0
    5.3326    2.7419   -1.3682 C   0  0  0  0  0  0  0  0  0  0  0  0
    6.8534    2.7540   -1.4046 C   0  0  0  0  0  0  0  0  0  0  0  0
    7.4725    3.9276   -2.1604 C   0  0  0  0  0  0  0  0  0  0  0  0
    6.7298    4.6749   -2.8073 O   0  0  0  0  0  0  0  0  0  0  0  0
    8.8644    4.1714   -2.0493 N   0  0  0  0  0  0  0  0  0  0  0  0
    9.7508    3.2262   -1.4318 C   0  0  0  0  0  0  0  0  0  0  0  0
   10.9328    3.5938   -0.7852 C   0  0  0  0  0  0  0  0  0  0  0  0
   11.9494    2.6439   -0.5990 C   0  0  0  0  0  0  0  0  0  0  0  0
   11.7132    1.2782   -0.8001 C   0  0  0  0  0  0  0  0  0  0  0  0
   12.8076    0.2336   -0.6111 C   0  0  0  0  0  0  0  0  0  0  0  0
   13.9207    0.4067   -1.6653 C   0  0  0  0  0  0  0  0  0  0  0  0
   13.3951    0.4009    0.7957 C   0  0  0  0  0  0  0  0  0  0  0  0
   12.2475   -1.1991   -0.7497 C   0  0  0  0  0  0  0  0  0  0  0  0
   10.4420    0.9065   -1.2118 C   0  0  0  0  0  0  0  0  0  0  0  0
    9.5599    1.8652   -1.6614 C   0  0  0  0  0  0  0  0  0  0  0  0
    9.4361    5.4152   -2.7371 C   0  0  3  0  0  0  0  0  0  0  0  0
    8.9519    6.6471   -1.8933 C   0  0  0  0  0  0  0  0  0  0  0  0
    9.7079    7.3819   -1.2456 O   0  0  0  0  0  0  0  0  0  0  0  0
    7.6139    6.9225   -1.9462 N   0  0  0  0  0  0  0  0  0  0  0  0
    6.9290    8.0242   -1.2727 C   0  0  0  0  0  0  0  0  0  0  0  0
    7.6008    9.3904   -1.4575 C   0  0  0  0  0  0  0  0  0  0  0  0
    7.6458    9.7408   -2.9357 C   0  0  0  0  0  0  0  0  0  0  0  0
    6.3233    9.7564   -3.4828 O   0  0  0  0  0  0  0  0  0  0  0  0
    5.6559    8.4987   -3.3513 C   0  0  0  0  0  0  0  0  0  0  0  0
    5.5261    8.0996   -1.8792 C   0  0  0  0  0  0  0  0  0  0  0  0
   10.9556    5.4706   -3.0660 C   0  0  0  0  0  0  0  0  0  0  0  0
   11.7130    6.6459   -2.9910 C   0  0  0  0  0  0  0  0  0  0  0  0
   13.0661    6.7146   -3.1014 N   0  0  0  0  0  0  0  0  0  0  0  0
   13.7278    5.5518   -3.2069 C   0  0  0  0  0  0  0  0  0  0  0  0
   13.0903    4.3358   -3.2870 C   0  0  0  0  0  0  0  0  0  0  0  0
   11.6983    4.3226   -3.3766 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.4719   -0.8541   -0.3978 H   0  0  0  0  0  0  0  0  0  0  0  0
    2.8570   -0.1379    1.0933 H   0  0  0  0  0  0  0  0  0  0  0  0
    2.6638   -1.2834   -1.6287 H   0  0  0  0  0  0  0  0  0  0  0  0
    2.6139   -2.1308   -0.2014 H   0  0  0  0  0  0  0  0  0  0  0  0
    2.7055    2.1050   -0.0754 H   0  0  0  0  0  0  0  0  0  0  0  0
    2.7409    1.2841   -1.6469 H   0  0  0  0  0  0  0  0  0  0  0  0
    4.9436    3.6071   -0.8294 H   0  0  0  0  0  0  0  0  0  0  0  0
    4.9377    2.7293   -2.3896 H   0  0  0  0  0  0  0  0  0  0  0  0
    7.0880    1.8488   -1.9609 H   0  0  0  0  0  0  0  0  0  0  0  0
    7.2683    2.7112   -0.3898 H   0  0  0  0  0  0  0  0  0  0  0  0
   11.1796    4.6371   -0.5967 H   0  0  0  0  0  0  0  0  0  0  0  0
```

(b)

```
     RDKit          3D

  8  7  0  0  1  0  0  0  0  0999 V2000
    7.1302   -4.3699   20.0222 C   0  0  0  0  0  0  0  0  0  0  0  0
    8.1668   -3.5169   18.8012 S   0  0  0  0  0  0  0  0  0  0  0  0
    9.6768   -4.3758   19.3596 C   0  0  0  0  0  0  0  0  0  0  0  0
   10.9103   -3.9921   18.5427 C   0  0  0  0  0  0  0  0  0  0  0  0
   10.6963   -4.2222   17.1227 N   0  0  0  0  0  0  0  0  0  0  0  0
   11.3542   -2.5873   18.7980 C   0  0  0  0  0  0  0  0  0  0  0  0
   10.6672   -1.6118   18.3983 O   0  0  0  0  0  0  0  0  0  0  0  0
   12.5295   -2.3583   19.5001 O   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0
  2  3  1  0
  3  4  1  0
  4  5  1  0
  4  6  1  0
  6  7  2  0
  6  8  1  0
M  END
$$$$
```

(c)

```
     RDKit          3D

 37 39  0  0  1  0  0  0  0  0999 V2000
   11.3457   -2.5751   18.7916 C   0  0  0  0  0  0  0  0  0  0  0  0
   10.6776   -1.6275   18.4048 O   0  0  0  0  0  0  0  0  0  0  0  0
   10.9094   -3.9877   18.5427 C   0  0  0  0  0  0  0  0  0  0  0  0
   10.6970   -4.2230   17.1233 N   0  0  0  0  0  0  0  0  0  0  0  0
    9.6771   -4.3746   19.3582 C   0  0  0  0  0  0  0  0  0  0  0  0
    8.1652   -3.5177   18.8011 S   0  0  0  0  0  0  0  0  0  0  0  0
    7.1298   -4.3704   20.0229 C   0  0  0  0  0  0  0  0  0  0  0  0
    6.7437   -5.7411   19.5005 C   0  0  0  0  0  0  0  0  0  0  0  0
    5.8330   -6.4143   20.4740 C   0  0  0  0  0  0  0  0  0  0  0  0
    6.3229   -7.2651   21.2645 O   0  0  0  0  0  0  0  0  0  0  0  0
    4.4380   -6.0613   20.5646 N   0  0  0  0  0  0  0  0  0  0  0  0
    3.7905   -5.3140   19.5123 C   0  0  0  0  0  0  0  0  0  0  0  0
    3.9808   -5.6719   18.1703 C   0  0  0  0  0  0  0  0  0  0  0  0
    3.3479   -4.9586   17.1500 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.4941   -3.8825   17.4485 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.8639   -3.0636   16.3378 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.8088   -1.9440   15.9452 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.5428   -2.4501   16.7770 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.5831   -3.9133   15.1074 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.3264   -3.5132   18.7943 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.9599   -4.2266   19.8139 C   0  0  0  0  0  0  0  0  0  0  0  0
    3.7041   -6.4847   21.7650 C   0  0  0  0  0  0  0  0  0  0  0  0
    3.3417   -5.3351   22.6608 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.1438   -4.9593   22.7679 O   0  0  0  0  0  0  0  0  0  0  0  0
    4.3897   -4.6514   23.3570 N   0  0  0  0  0  0  0  0  0  0  0  0
    4.2522   -3.4469   24.1745 C   0  0  0  0  0  0  0  0  0  0  0  0
    3.4035   -3.6738   25.4332 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.9401   -3.3043   25.1853 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.8519   -1.9932   24.6895 O   0  0  0  0  0  0  0  0  0  0  0  0
    2.3149   -2.0466   23.3647 C   0  0  0  0  0  0  0  0  0  0  0  0
    3.8334   -2.2214   23.3515 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.5932   -7.4522   21.4648 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.4596   -7.0756   20.7317 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.4790   -7.9820   20.4737 N   0  0  0  0  0  0  0  0  0  0  0  0
    0.5649   -9.2668   20.9117 C   0  0  0  0  0  0  0  0  0  0  0  0
    1.6781   -9.6842   21.6443 C   0  0  0  0  0  0  0  0  0  0  0  0
    2.6953   -8.7688   21.9192 C   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  2  0
  1  3  1  0
  3  4  1  0
  3  5  1  0
  5  6  1  0
  6  7  1  0
  7  8  1  0
  8  9  1  0
```

(d)

```
ATOM      1  C01 LIG A 145       9.677  -4.375  19.358  0.00  0.00           C
ATOM      2  S   LIG A 145       8.165  -3.518  18.801  0.00  0.00           S
ATOM      3  C02 LIG A 145       7.130  -4.370  20.023  0.00  0.00           C
ATOM      4  C03 LIG A 145       6.744  -5.741  19.500  0.00  0.00           C
ATOM      5  C04 LIG A 145       5.833  -6.414  20.474  0.00  0.00           C
ATOM      6  O05 LIG A 145       6.323  -7.265  21.264  0.00  0.00           O
ATOM      7  N06 LIG A 145       4.438  -6.061  20.565  0.00  0.00           N
ATOM      8  C07 LIG A 145       3.790  -5.314  19.512  0.00  0.00           C
ATOM      9  C08 LIG A 145       3.981  -5.672  18.170  0.00  0.00           C
ATOM     10  C09 LIG A 145       3.348  -4.959  17.150  0.00  0.00           C
ATOM     11  C10 LIG A 145       2.494  -3.882  17.448  0.00  0.00           C
ATOM     12  C11 LIG A 145       1.864  -3.064  16.338  0.00  0.00           C
ATOM     13  C12 LIG A 145       2.809  -1.944  15.945  0.00  0.00           C
ATOM     14  C13 LIG A 145       0.543  -2.450  16.777  0.00  0.00           C
ATOM     15  C14 LIG A 145       1.583  -3.913  15.107  0.00  0.00           C
ATOM     16  C15 LIG A 145       2.326  -3.513  18.794  0.00  0.00           C
ATOM     17  C16 LIG A 145       2.960  -4.227  19.814  0.00  0.00           C
ATOM     18  C17 LIG A 145       3.704  -6.485  21.765  0.00  0.00           C
ATOM     19  C18 LIG A 145       3.342  -5.335  22.661  0.00  0.00           C
ATOM     20  O19 LIG A 145       2.144  -4.959  22.768  0.00  0.00           O
ATOM     21  N20 LIG A 145       4.390  -4.651  23.357  0.00  0.00           N
ATOM     22  C21 LIG A 145       4.252  -3.447  24.174  0.00  0.00           C
ATOM     23  C22 LIG A 145       3.404  -3.674  25.433  0.00  0.00           C
ATOM     24  C23 LIG A 145       1.940  -3.304  25.185  0.00  0.00           C
ATOM     25  O   LIG A 145       1.852  -1.993  24.690  0.00  0.00           O
ATOM     26  C24 LIG A 145       2.315  -2.047  23.365  0.00  0.00           C
ATOM     27  C25 LIG A 145       3.833  -2.221  23.351  0.00  0.00           C
ATOM     28  C26 LIG A 145       2.593  -7.452  21.465  0.00  0.00           C
ATOM     29  C27 LIG A 145       1.460  -7.076  20.732  0.00  0.00           C
ATOM     30  N   LIG A 145       0.479  -7.982  20.474  0.00  0.00           N
ATOM     31  C28 LIG A 145       0.565  -9.267  20.912  0.00  0.00           C
ATOM     32  C29 LIG A 145       1.678  -9.684  21.644  0.00  0.00           C
ATOM     33  C   LIG A 145       2.695  -8.769  21.919  0.00  0.00           C
END
```

(e)

```
REMARK  9 active torsions:
REMARK  status: ('A' for Active; 'I' for Inactive)
REMARK    1  A    between atoms: S_2  and  C02_3
REMARK    2  A    between atoms: C02_3  and  C03_4
REMARK    3  A    between atoms: C03_4  and  C04_5
REMARK       I    between atoms: C04_5  and  N06_7
REMARK    4  A    between atoms: N06_7  and  C07_8
REMARK    5  A    between atoms: N06_7  and  C17_18
REMARK    6  A    between atoms: C10_11  and  C11_12
REMARK    7  A    between atoms: C17_18  and  C26_28
REMARK    8  A    between atoms: C17_18  and  C18_19
REMARK       I    between atoms: C18_19  and  N20_21
REMARK    9  A    between atoms: N20_21  and  C21_22
ROOT
ATOM      1  C01 LIG A 145       9.677  -4.375  19.358  0.00  0.00     0.069 C
ATOM      2  S   LIG A 145       8.165  -3.518  18.801  0.00  0.00    -0.139 SA
ENDROOT
BRANCH   2   3
ATOM      3  C02 LIG A 145       7.130  -4.370  20.023  0.00  0.00     0.067 C
BRANCH   3   4
ATOM      4  C03 LIG A 145       6.744  -5.741  19.500  0.00  0.00     0.100 C
BRANCH   4   5
ATOM      5  C04 LIG A 145       5.833  -6.414  20.474  0.00  0.00     0.239 C
ATOM      6  O05 LIG A 145       6.323  -7.265  21.264  0.00  0.00    -0.272 OA
ATOM      7  N06 LIG A 145       4.438  -6.061  20.565  0.00  0.00    -0.289 N
BRANCH   7   8
ATOM      8  C07 LIG A 145       3.790  -5.314  19.512  0.00  0.00     0.063 A
ATOM      9  C16 LIG A 145       2.960  -4.227  19.814  0.00  0.00     0.017 A
ATOM     10  C15 LIG A 145       2.326  -3.513  18.794  0.00  0.00    -0.003 A
ATOM     11  C10 LIG A 145       2.494  -3.882  17.448  0.00  0.00    -0.023 A
ATOM     12  C09 LIG A 145       3.348  -4.959  17.150  0.00  0.00    -0.003 A
ATOM     13  C08 LIG A 145       3.981  -5.672  18.170  0.00  0.00     0.017 A
BRANCH  11  14
ATOM     14  C11 LIG A 145       1.864  -3.064  16.338  0.00  0.00     0.019 C
ATOM     15  C14 LIG A 145       1.583  -3.913  15.107  0.00  0.00     0.004 C
ATOM     16  C13 LIG A 145       0.543  -2.450  16.777  0.00  0.00     0.004 C
ATOM     17  C12 LIG A 145       2.809  -1.944  15.945  0.00  0.00     0.004 C
ENDBRANCH  11  14
ENDBRANCH   7   8
BRANCH   7  18
ATOM     18  C17 LIG A 145       3.704  -6.485  21.765  0.00  0.00     0.193 C
BRANCH  18  19
ATOM     19  C26 LIG A 145       2.593  -7.452  21.465  0.00  0.00     0.027 A
ATOM     20  C27 LIG A 145       1.460  -7.076  20.732  0.00  0.00     0.097 A
ATOM     21  N   LIG A 145       0.479  -7.982  20.474  0.00  0.00    -0.243 NA
ATOM     22  C28 LIG A 145       0.565  -9.267  20.912  0.00  0.00     0.099 A
ATOM     23  C29 LIG A 145       1.678  -9.684  21.644  0.00  0.00     0.021 A
ATOM     24  C   LIG A 145       2.695  -8.769  21.919  0.00  0.00     0.001 A
ENDBRANCH  18  19
BRANCH  18  25
```

(f)

```
ROOT
ATOM      4  C03 LIG A 145       6.399  -3.349  20.875  0.00  0.00     0.100 C
ATOM      5  C04 LIG A 145       7.126  -3.158  22.166  0.00  0.00     0.239 C
ATOM      6  O05 LIG A 145       8.251  -2.589  22.143  0.00  0.00    -0.272 OA
ATOM      7  N06 LIG A 145       6.539  -3.548  23.424  0.00  0.00    -0.289 N
ATOM      8  C07 LIG A 145       5.403  -4.439  23.469  0.00  0.00     0.063 A
ATOM      9  C16 LIG A 145       5.584  -5.828  23.467  0.00  0.00     0.017 A
ATOM     10  C15 LIG A 145       4.483  -6.688  23.500  0.00  0.00    -0.003 A
ATOM     11  C10 LIG A 145       3.174  -6.177  23.518  0.00  0.00    -0.023 A
ATOM     12  C09 LIG A 145       2.999  -4.782  23.545  0.00  0.00    -0.003 A
ATOM     13  C08 LIG A 145       4.100  -3.923  23.512  0.00  0.00     0.017 A
ATOM     14  C11 LIG A 145       1.983  -7.108  23.626  0.00  0.00     0.019 C
ATOM     15  C14 LIG A 145       2.336  -8.386  24.373  0.00  0.00     0.004 C
ATOM     16  C13 LIG A 145       0.827  -6.454  24.368  0.00  0.00     0.004 C
ATOM     17  C12 LIG A 145       1.511  -7.480  22.233  0.00  0.00     0.004 C
ATOM     18  C17 LIG A 145       7.134  -3.005  24.652  0.00  0.00     0.193 C
ATOM     19  C26 LIG A 145       8.509  -3.546  24.930  0.00  0.00     0.027 A
ATOM     20  C27 LIG A 145       9.661  -2.771  24.750  0.00  0.00     0.097 A
ATOM     21  N   LIG A 145      10.888  -3.293  25.019  0.00  0.00    -0.243 NA
ATOM     22  C28 LIG A 145      11.034  -4.569  25.468  0.00  0.00     0.099 A
ATOM     23  C29 LIG A 145       9.911  -5.375  25.663  0.00  0.00     0.021 A
ATOM     24  C   LIG A 145       8.644  -4.857  25.392  0.00  0.00     0.001 A
ATOM     25  C18 LIG A 145       6.182  -3.019  25.815  0.00  0.00     0.277 C
ATOM     26  O19 LIG A 145       5.645  -1.952  26.215  0.00  0.00    -0.268 OA
ATOM     27  N20 LIG A 145       5.932  -4.261  26.481  0.00  0.00    -0.238 NA
ATOM     28  C21 LIG A 145       5.137  -4.455  27.692  0.00  0.00     0.111 C
ATOM     29  C22 LIG A 145       5.720  -5.526  28.624  0.00  0.00     0.057 C
ATOM     30  C23 LIG A 145       5.085  -6.893  28.357  0.00  0.00     0.144 C
ATOM     31  O   LIG A 145       3.687  -6.797  28.433  0.00  0.00    -0.354 OA
ATOM     32  C24 LIG A 145       3.281  -6.112  27.276  0.00  0.00     0.144 C
ATOM     33  C25 LIG A 145       3.642  -4.632  27.394  0.00  0.00     0.057 C
ENDROOT
TORSDOF 9
```

Figure N1: Sample files for c3D-DQN methods from steps **(5)** to **(8)** in Figure 4.4 where **(a)** generation of SDF file from compound-amino acid adduct (partial SDF shown), **(b)** cysteine residue SDF file from the prepared protein PDBQT file, **(c)** aligned SDF file from **(b)** based on maximum common substructure match (partial SDF file shown), **(d)** PDB file generated from SDF file using PyMOL, **(e)** PDBQT file for constrained docking generated using prepare_ligand4.py in AutoDockTools (partial PDBQT file shown), and **(f)** Output PDBQT file from **(e)** to get processed to prepare for score_only docking using AutoDock Vina.