To explore drug space smarter: artificial intelligence in drug design for G protein-coupled receptors

Liu, X.

Citation

Liu, X. (2022, February 15). *To explore drug space smarter: artificial intelligence in drug design for G protein-coupled receptors*. Retrieved from https://hdl.handle.net/1887/3274010

Version: Publisher's Version

License: Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden

Downloaded from: https://hdl.handle.net/1887/3274010

Note: To cite this publication please use the final published version (if applicable).

To explore drug space smarter

Artificial intelligence in drug design for G protein-coupled receptors

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op 15 februari 2022
klokke 10:00 uur

door

Xuhan Liu

刘 许晗

geboren te Henan, China
in 1989

Promotores:

Prof. dr. Adriaan P. IJzerman

Prof. dr. Gerard J. P. van Westen

Prof. dr. Kai Ye

Promotiecommissie:

Prof. dr. Hubertus Irth

Prof. dr. Joke Bouwstra

Prof. dr. Mario van der Stelt

Prof. dr. Iwan J.P. de Esch (Vrije Universiteit Amsterdam)

Dr. Francesca Grisoni (Technische Universiteit Eindhoven)

Copyright © 2022 Xuhan Liu All rights reserved

Thesis lay-out & Cover design: Xuhan Liu

Printing: PrintSupport4U

ISBN: 978-94-92597-94-6

This research is financially supported by the China Scholarship Council (CSC), CSC No. 201709110143

The research described in this thesis was performed at the Division of Drug Discovery and Safety of the Leiden Academic Centre for Drug Research (LACDR), Leiden University (Leiden, The Netherlands).

Content

1. General introduction 1

1.1. AI in drug discovery 4

1.2. G protein-coupled receptors 8

1.3. Adenosine receptors 9

1.4. Research questions 9

1.5. Thesis outline 10

2. Computational approaches for *de novo* drug design: past, present and future 17

2.1. Introduction 19

2.2. *De novo* drug design 21

2.2.1. Molecular representations 22

2.2.2. Multiple objectives 23

2.3. Optimization methods 24

2.3.1. Evolutionary algorithms 25

2.3.2. Particle swarm optimization 28

2.3.3. Simulated annealing 29

2.4. Deep learning algorithms 30

2.4.1. Recurrent neural networks 30

2.4.2. Variational autoencoders 33

2.4.3. Deep reinforcement learning 35

2.4.4. Generative adversarial networks 37

2.5. Competition or cooperation? 40

2.6. Conclusion and perspective 42

3. An exploration strategy improves the diversity of *de novo* ligands using deep reinforcement learning: a case for the adenosine A_{2A} receptor 49

3.1. Introduction 51

3.2. Dataset and methods 52

3.2.1. Data source 52

3.2.2. Prediction model (QSAR) 53

3.2.3. Generative model 53

3.2.4. Reinforcement learning 56

3.2.5. Exploration strategy 56

3.2.6. Molecular diversity 57

3.3. Results and discussion 58

3.3.1. Performance of predictors 58

3.3.2. SMILES syntax learning 58

3.3.3. Conditional SMILES generation 60

3.3.4. Exploration strategy 62

3.3.5. Comparison with other methods 64

3.4. Conclusion and future prospect 71

4. *DrugEx* v2: *de novo* design of drug molecules by Pareto-based multi-objective reinforcement learning in polypharmacology 81

4.1. Introduction 83

4.2. Materials and methods 85

4.2.1. Data source 85

4.2.2. Prediction model 85

4.2.3. Generative model 87

4.2.4. Reinforcement learning 88

4.2.5. Algorithm extrapolation 91

4.2.6. Exploration strategy 92

4.2.7. Molecular diversity 93

4.3. Results and discussion 94

4.3.1. Performance of predictors 94

4.3.2. Model optimization 95

4.3.3. Performance comparisons 96

4.4. Conclusion and future prospect 102

5. *DrugEx v3*: scaffold-constrained drug design with graph Transformer-based reinforcement learning 113

5.1. Introduction 115

5.2. Materials and methods 116

5.2.1. Data source 116

5.2.2. Molecular representations 118

5.2.3. End-to-end deep learning 119

5.2.4. Multi-objective optimization 122

5.2.5. Reinforcement learning 124

5.2.6. Performance evaluation 125

5.3. Results and discussion 126

5.3.1. Fragmentation of molecule 126

5.3.2. Pre-training & fine-tuning 127

5.3.3. Policy gradient 131

5.3.4. Generated molecules 134

5.4. Conclusion and Future Perspective 135

6. GenUI: interactive and extensible open source software platform for *de novo* molecular generation and cheminformatics 143

6.1. Introduction 145

6.2. Software architecture 148

6.3. Frontend 150

6.3.1. Graphical user interface (GUI) 150

6.3.2. Projects 151

6.3.3. Compounds 151

6.3.4. QSAR models 151

6.3.5. Generators 153

6.3.6. Maps 157

6.3.7. JavaScript API 158

6.3.8. Model components 159

6.3.9. REST API components 160

6.4. Backend 161

6.4.1. Python API 161

6.4.2. Extensions 162

6.4.3. Automatic code discovery 163

6.4.4. Generic views and viewsets 163

6.4.5. Asynchronous tasks 164

6.4.6. Integration of new features with the two APIs 165

6.4.7. Compounds import 165

6.4.8. QSAR models 165

6.4.9. Molecular generators 166

6.4.10. Chemical space maps 167

6.5. Deployment 167
6.6. Future directions 168
6.7. Conclusions 169

7. Conclusions and future perspectives 177

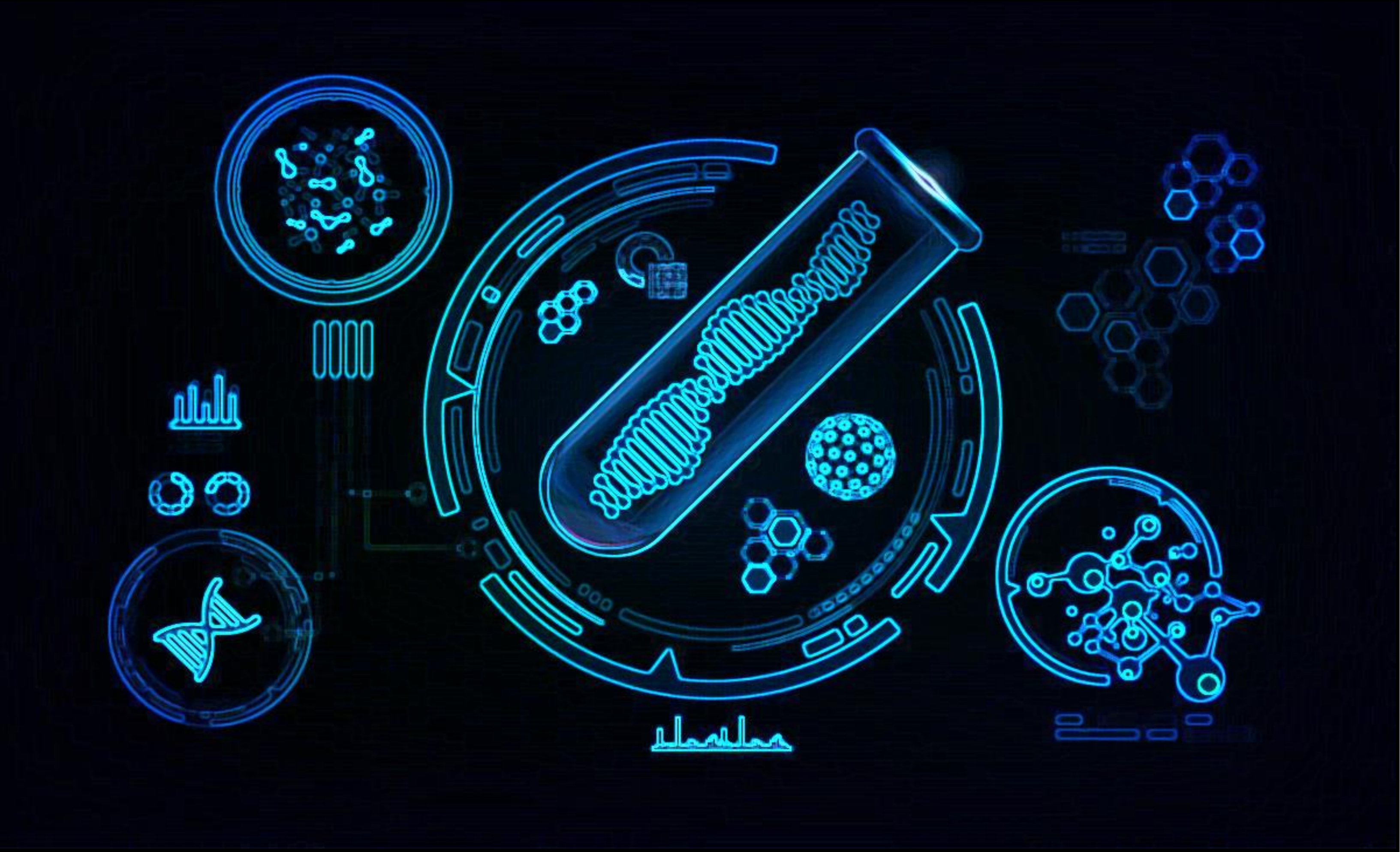
7.1. Conclusions 179
7.2. Further perspectives 182
  7.2.1. New AI technologies 182
  7.2.2. Different constraint conditions 183
  7.2.3. Designing various kind of molecules 184
7.3. Final notes 185

Appendix 191

Summary 193
Samenvatting 195
中文总结 199
List of publications 201
Curriculum Vitae 203
Acknowledgements 205

Chapter 1

General introduction



About this thesis

Drug discovery is a time- and resource-consuming process; from original idea to the regulatory approval of the finished product tends to take more than 10 years and bring costs in excess of $1 billion [1]. In order to decrease the cost and save time in this process, a plethora of computational methods have been developed [2,3]. Benefiting from the rapid growth of high throughput technologies, modern pharmacology has become a data-rich field with high-dimensional biological data accumulated in public databases, e.g. compound profiling data, gene expression data, transgenic phenotyping data, proteomics data, publications and patent information, *etc.* [4]. In order to analyze these ‘big data’ [5], artificial intelligence (AI) approaches are increasingly exploited in diverse scenarios to accelerate drug discovery [6].

AI is defined as the simulation of human intelligence to make machines think like humans and mimic their actions such as learning, reasoning, perception, and problem solving [7]. A typical subset of AI is machine learning (ML), which is trained on a large amount of (curated) data, and used to make predictions or decisions without being explicitly programmed [8]. One of the most popular ML methods are neural networks which consist of multiple layers of neurons (input, hidden, and output layers) that are mimics of neural activity in the human brain. Recently, neural networks have become extremely ‘deep’ because more and more hidden layers are added and organized with different architectures [9]. These so-called ‘deep learning’ methods have achieved major breakthroughs in e.g., image recognition, natural language processing, decision making, and other data-rich domains [10].

With the development of deep learning and rapid advances made in computational hardware, AI has further expanded its application scope in drug discovery [11]. In this thesis, I will introduce some of my projects about the application of AI in *de novo* drug design for G protein-coupled receptors and discuss its possible role in the future.

1.1. AI in drug discovery

The motivation behind launching a drug discovery project typically is that there is no or in some form limited suitable cure available to treat an existing disease or to meet clinical needs. The process of drug discovery can be seen as ‘serendipity’, *i.e.* it is about searching the optimal molecules from a huge chemical space (comprised of 10^{33}~10^{60} ‘drug-like’ molecules) [12]. In order to find drug candidates that are efficacious, safe, and meet clinical and commercial needs, the whole process of drug discovery includes target identification and validation, compound screening, lead optimization, preclinical development and finally the selection of drug candidates for clinical trials (Fig. 1.1) [13-15]. In all stages of drug discovery and development, AI algorithms are being developed and utilized in many aspects, such as understanding disease mechanisms to identify novel targets, providing target-disease associations, predicting properties of compounds, lead compound design and optimization, developing new biomarkers for prognosis, analyzing biometric and other data from wearable patient monitoring devices [14].

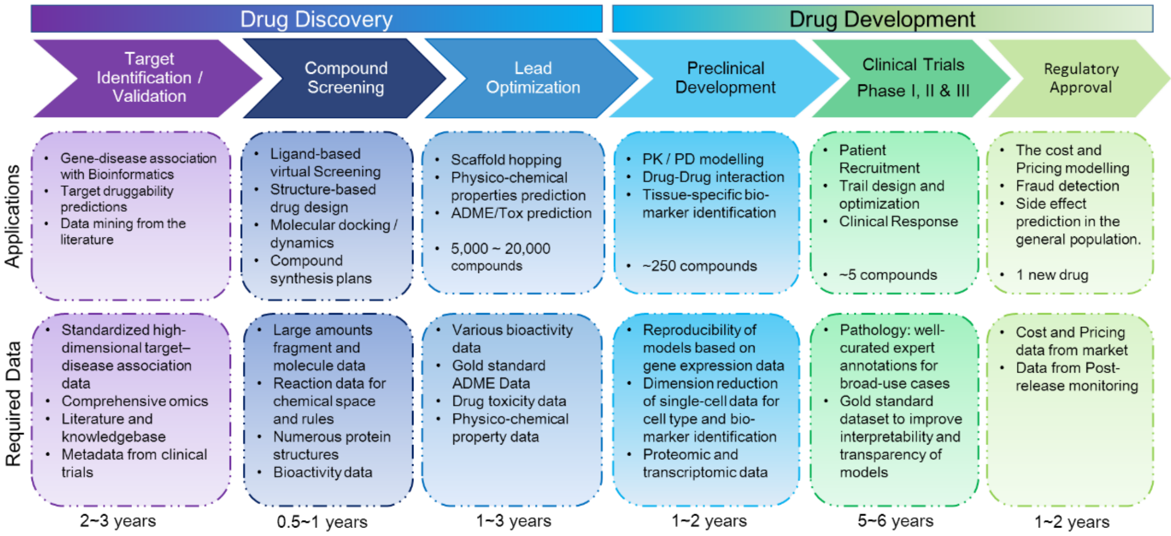

Drug Discovery
Drug Development
Target Identification / Validation
Compound Screening
Lead Optimization
Preclinical Development
Clinical Trials Phase I, II & III
Regulatory Approval
Applications
• Gene-disease association with Bioinformatics
• Target druggability predictions
• Data mining from the literature
• Ligand-based virtual Screening
• Structure-based drug design
• Molecular docking / dynamics
• Compound synthesis plans
• Scaffold hopping
• Physico-chemical properties prediction
• ADME/Tox prediction
• 5,000 ~ 20,000 compounds
• PK / PD modelling
• Drug-Drug interaction
• Tissue-specific bio-marker identification
• ~250 compounds
• Patient Recruitment
• Trail design and optimization
• Clinical Response
• ~5 compounds
• The cost and Pricing modelling
• Fraud detection
• Side effect prediction in the general population.
• 1 new drug
Required Data
• Standardized high-dimensional target–disease association data
• Comprehensive omics
• Literature and knowledgebase
• Metadata from clinical trials
• Large amounts fragment and molecule data
• Reaction data for chemical space and rules
• Numerous protein structures
• Bioactivity data
• Various bioactivity data
• Gold standard ADME Data
• Drug toxicity data
• Physico-chemical property data
• Reproducibility of models based on gene expression data
• Dimension reduction of single-cell data for cell type and bio-marker identification
• Proteomic and transcriptomic data
• Pathology: well-curated expert annotations for broad-use cases
• Gold standard dataset to improve interpretability and transparency of models
• Cost and Pricing data from market
• Data from Post-release monitoring
2~3 years
0.5~1 years
1~3 years
1~2 years
5~6 years
1~2 years

Fig. 1.1: Applications of AI in the pipeline of drug discovery and the required data. The information in this figure is collected from Ref. [14,15].

The initial phase of drug discovery starts from developing a hypothesis that the candidate drugs will lead to a therapeutic effect in a disease state when a protein or pathway is the inhibited or activated. Therefore the first and foremost step in developing a novel drug is target identification and validation. AI approaches are extensively applied in gene-disease association modelling. For example, mRNA/protein levels could be examined to determine if they are expressed differently in disease and it thus can reflect whether they are correlated with the exacerbation or progression of diseases [16]. In addition, data mining and meta-analysis have resulted in a significant increase in target identification with literature and knowledge bases that contain successful and failed trails [17,18].

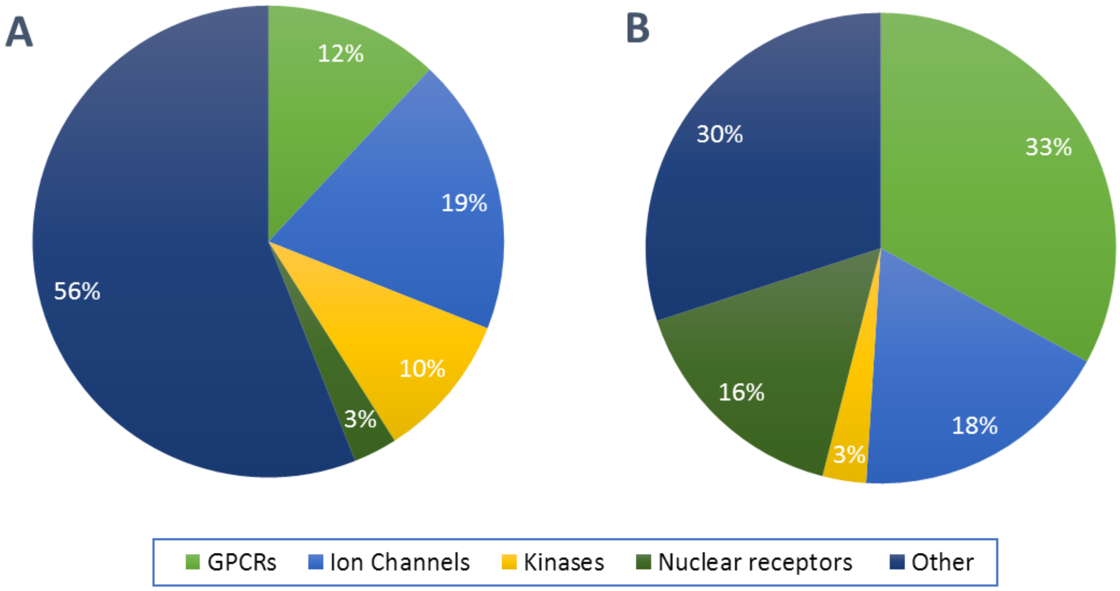


Fig. 1.2: The percentage of human drug targets in major protein families (A) and the proportion of small-molecule drugs targeting these families in human (B). The figure is adapted from Ref. [19].

After being identified, the role of these targets in a disease is validated using physiologically relevant in vitro/vivo models. An important question is how likely it is that a (small molecule) drug can be developed for the selected target, *i.e.* these proteins should bind small molecules as potential drugs. Through systems biology analysis druggable proteins have also been found in protein-protein interaction networks and these tend to be highly connected. For further investigations information on the 3D structure of the targets is very useful and can be obtained through X-ray crystallography [20], nuclear magnetic resonance spectroscopy [21] or cryogenic electron microscopy [22] technologies. It is worth mentioning that AI methods have recently been shown to be key in the prediction of 3D structure. For example, the DeepMind team exploited deep learning models to develop the AlphaFold algorithm which has dramatically improved the accuracy of protein structure prediction [23]. It is now known that certain target classes are more amenable to small molecule drug discovery (Fig. 1.2), such as G protein-coupled receptors (GPCR), ion channels, kinases, *etc*. [19]. Despite the ultimate validation of the target in later clinical trials, target validation in the early phage is essential to make efforts on the completion of promising projects.

After the determination of the target, an appropriate organic small molecule library will be screened to search for feasible compounds as “hit compounds” that can be optimized into “drug leads” that can inhibit or activate the given target after binding. Besides random/trial-and-error experimental high throughput screening (HTS) technologies necessitating complex laboratory automation, virtual screening has also become a common paradigm to discover drug hits [24]. With structural and bioactivity data being accumulated, the quantitative relationship between structure and activity (QSAR) can be modelled with ML methods to predict the activity of the given molecules rapidly. In order to improve the application scope and the accuracy of ligand-based virtual screening, van Westen et al. proposed Proteochemometric modelling (PCM), updating classical QSAR models with target information, which was able to deal with different protein targets within one model [25,26]. In addition, deep learning can increase the performance substantially, when predicting the properties and activities of small organic molecules [27]. If the 3D structure of a protein target is available, molecular docking is also an effective method for structure-based virtual screening, which allows for the analysis of the ligand-target interaction by minimizing objective functions [28]. In this way it can provide a more mechanistic explanation of the interaction between ligand and target. Furthermore, molecular dynamics is applied as a computational method to study the dynamic mechanism of the drug-target interaction by calculating continuous motions using force fields [29].

For virtual screening, compound libraries need to be assembled containing small organic molecules that obey chemical guidelines such as the Lipinski Rule of Five [30]. The molecules in the library can be collected from public databases, such as ChEMBL [31], ZINC [32], PubChem [33], *etc.*. On the other hand it is also common strategy to *de novo* design drug molecules with computational methods. This direction is my major study and I will give a detailed description of AI-based approaches applied in drug *de novo* design and make a comparison between different categories of methods in the next chapter. In order to run an experimental assay after virtual compound screening or *de novo* design, these drug leads are required to be synthesized in reality. AI algorithms can also be used to predict chemical synthesis routes which is comprised of "reversed" reactions decomposing the molecules (retrosynthesis). For example, Segler et al. scored the tree nodes in conjunction with deep learning and Monte Carlo tree search (MCTS) to search for the most promising synthetic pathways [34].

The selected drug leads need to be optimized to maintain favorable properties while eliminating deficiencies in the lead structure. In other words, drug discovery is a classical multi-objective optimization problem. The lead structure can be optimized either by changing or adding substituents through rational drug design or modifying the basic scaffold via scaffold hopping to avoid patent conflicts. In addition to the efficacy of drug leads, safety is a second important aspect to be persecuted. Typically, absorption, distribution, metabolism and excretion (ADME) are common metrics to evaluate drug safety. The principles of ADME are also important parameters in pharmacokinetics / pharmacodynamics (PK/PD) modelling, which is another approach to determine the toxicity and the drug dose required for the desired effect [35]. Side effects of drugs are often caused by binding to unwanted targets. For example, hERG (human Ether-à-go-go-Related Gene), a potassium ion channel, has an inclination to bind many drug molecules because of its large ligand binding pocket [36]. It may result in long QT syndrome (a change to the heart rhythm that can lead to fast chaotic heartbeats) when hERG is inhibited by potential drug candidates [37].

In the later stage of drug development, AI-based approaches for biomarker discovery have demonstrated to lead to a better understanding of the molecular mechanisms of drug effects and to identify the right drug for the right patients [38]. It will be most beneficial to build and validate such AI models on datasets collected in the preclinical stage [39]. After being validated using independent datasets, corresponding biomarkers can be applied to stratify patients and identify potential indications [14]. In spite of the successful use of AI in early drug discovery and development there are a number of key issues that still need to be further addressed. Because they lack transparency, ML methods are often called ‘black-box’ approaches, which sparked criticism from end-users in the clinical adoption [40]. Interpretability is indeed a general weakness of ML-based predictive models [41]. One of the other key issues is the generalizability, namely that models need to be validated in the context of multi-site, multi-institutional datasets to prove the robustness of their predictions beyond the original training set. Faced with these challenges in drug discovery, the scientific community has nevertheless been making great contributions, including model training with parameter optimization approaches [42], interpretation of prediction results and deduction of their biological insights [43], and model reproducibility [44].

1.2. G protein-coupled receptors

G protein-coupled receptors (GPCRs), containing a characteristic structure of seven trans-membrane helices, are the largest family of cell-surface receptors. The superfamily consists of almost 900 members encoded by approximately 4% of human genes, which are classified into five families: rhodopsin, secretin, glutamate, adhesion and frizzled/Taste2 [45]. When binding signaling molecules from the extracellular environment, such as hormones or neurotransmitters, GPCRs will be activated. This activation will cause a conformational change and in turn activate the associated G protein by exchanging bound GDP for GTP [46]. Subsequently, α subunit of the G protein, bound with GTP, dissociates from the β and γ subunits to further regulate intracellular downstream biological pathways [46]. GPCRs are important switches to determine on-off states of numerous signaling pathways and they are involved in many biological processes, such as cell survival, proliferation, motility, *etc.* [47]. The aberrant activity or expression of certain GPCRs also contributes to some severe diseases, such as type II diabetes, Alzheimer’s disease, hypertension, and heart failure [48]. In addition, the role of GPCRs in tumor growth, progression, and metastasis formation has become more apparent. Therefore, approximately 34% of all FDA approved drugs have a GPCRs as drug target (Fig. 1.2) [48].

1.3. Adenosine receptors

Adenosine receptors (ARs) are a class of purinergic G protein-coupled receptors with adenosine as the endogenous ligand. Adenosine is an essential component for all of life because it is one of the four precursors to nucleic acids and one of its derivatives (*i.e.* ATP) is the "molecular unit of currency" for energy transfer in metabolic processes. In addition, adenosine can also form a signaling molecule (*i.e.* cAMP) to modulate a variety of biological pathways which are activated by the adenosine receptors when binding to adenosine. There are four subtypes of adenosine receptors namely A_1, A_{2A}, A_{2B} and A_3 which are widely distributed in human tissues and have been implicated in many physiological and pathological functions [49]. These dysfunctions include lipolysis, cardiac rhythm and circulation, immune function, renal blood flow, sleep regulation and angiogenesis, as well as inflammatory diseases, neurodegenerative disorders and ischemia–reperfusion damage [50].

As promising therapeutic targets ARs have been studied for a long time. First of all, adenosine itself can act as an agonist for the treatment of supraventricular tachycardia [51]. In addition, caffeine is the most commonly consumed “drug” in the world, but this AR antagonist is also used for treating apnoea in premature infants [52]. Interestingly, an inverse correlation between caffeine consumption and risk of Parkinson’s disease has been demonstrated [53]. However there are some serious challenges when taking ARs as drug targets due to their ubiquitous expression and the complexity of adenosine signaling throughout the human body. Tissue- and target-specific issues should be carefully taken into consideration [50]. Due to the complexity of adenosine signaling, it is crucial to understand the disease process when ARs are targeted in different cellular elements and disease courses

1.4. Research questions

The aim of this thesis is to apply AI approaches to increase the efficiency of the drug *de* *novo* design process and thereby decrease cost and save time. Faced with the complexity of drug discovery and rapid development of AI technologies, I will investigate the following research questions which will be addressed in the following chapters:

1) Can AI support *de novo* drug design reliably and suggest active molecules for a single target as drug candidates?
2) Can AI be adjusted to polypharmacology to design desired molecules that bind multiple targets in order to balance efficacy and safety?
3) Can we improve the generality of AI models to design active molecules based on user-provided information such as fragments?
4) How to make these computational methods easily accessible by users who are not experts in computer coding skills?

1.5. Thesis outline

Up to now, there are numerous computational methods available for *de novo* drug design. In **Chapter 2**, I give a systematic overview of these methods, including optimization methods and deep learning methods. I conclude with describing the advantages and disadvantages of these methods and propose some possibilities of their combinations.

In **Chapter 3**, I introduce the first version of my proposed method, ***DrugEx***, which is a deep learning-based model for *de novo* drug design. In this method, a recurrent neural network was implemented to construct the generator. During the training process, the generator acted as the agent and the predictor acted as the environment interplay under the reinforcement learning framework. Here we add another pre-trained generator as the exploration strategy to improve the diversity of generated molecules. To evaluate the performance of my proposed method, the $A_{2A}AR$ is taken as an example, and most of the generated molecules are presumed to be active and located in the same region of chemical space occupied by ligands in the training set.

In **Chapter 4**, ***DrugEx*** was updated to the second version to include polypharmacology. In the first version it exclusively dealt with a single objective, while this second version has the ability to deal with multiple objectives. The concept of evolutionary algorithms was merged into our method such that *crossover* and *mutation* operations were implemented by the same deep learning model of the *agent* to update the exploration strategy. In this chapter, three protein targets were chosen for the case study (two adenosine receptors, A_1AR and $A_{2A}AR$, and hERG in this study). Scores for all objectives provided by the *environment* constructed by three predictors were used for constructing Pareto ranks of the generated molecules with non-dominated sorting and Tanimoto-based crowding distance algorithms. The results demonstrate a generation of compounds with a diverse predicted selectivity profile toward multiple targets, offering the potential of high efficacy and lower toxicity.

In the first two versions our proposed method can handle multiple objectives, but these objectives have to be fixed during the training process. In other words, if the objectives are changed, the model has to be retrained. In **Chapter 5**, ***DrugEx*** was updated again for scaffold-constrained molecule generation. Here, we use end-to-end deep learning methods to implement the new ***DrugEx*** model, which has the ability to generate molecules containing pharmacological the scaffold containing multiple fragments given by users. We also trained the generator into reinforcement learning framework to ensure that the generated molecules are most likely active towards $A_{2A}AR$.

In order to access the abovementioned methods conveniently, we developed a web-based online toolkit. In **Chapter 6**, a detailed description of this toolkit is provided, which constitutes a graphic utility interface named ***GenUI***. It contains two main parts: client and backend components. In the backend components, we use Docker to make the installation automatically without complex configuration. In addition, it contains managers to distribute the computational resources and dispatch different tasks from different users in the task queue. In the client components, users can easily create their project and schedule a variety of tasks automatically, including data collection and preprocessing, QSAR modelling, Generator training, novel molecule design and chemical space visualization. The generated data for each user is stored in the server, and users can easily download it into a local machine.

Finally, I will draw general conclusions in **Chapter 7**. Moreover, I will also put forward some key points for future directions and expected trends regarding computational methods in *de novo* drug design.

References

1. Paul SM, Mytelka DS, Dunwiddie CT, Persinger CC, Munos BH, Lindborg SR, Schacht AL (2010) How to improve R&D productivity: the pharmaceutical industry's grand challenge. Nat Rev Drug Discov 9 (3):203-214. doi:10.1038/nrd3078
2. Liu X, IJzerman AP, van Westen GJP (2021) Computational Approaches for De Novo Drug Design: Past, Present, and Future. Methods Mol Biol 2190:139-165. doi:10.1007/978-1-0716-0826-5_6
3. Schneider G, Fechner U (2005) Computer-based *de novo* design of drug-like molecules. Nat Rev Drug Discov 4 (8):649-663. doi:10.1038/nrd1799
4. Macarron R, Banks MN, Bojanic D, Burns DJ, Cirovic DA, Garyantes T, Green DV, Hertzberg RP, Janzen WP, Paslay JW, Schopfer U, Sittampalam GS (2011) Impact of high-throughput screening in biomedical research. Nat Rev Drug Discov 10 (3):188-195. doi:10.1038/nrd3368
5. Howe D, Costanzo M, Fey P, Gojobori T, Hannick L, Hide W, Hill DP, Kania R, Schaeffer M, St Pierre S, Twigger S, White O, Rhee SY (2008) Big data: The future of biocuration. Nature 455 (7209):47-50. doi:10.1038/455047a
6. Fleming N (2018) How artificial intelligence is changing drug discovery. Nature 557 (7707):S55-S57. doi:10.1038/d41586-018-05267-x
7. Rajaram NS (1990) Artificial intelligence: a technology review. ISA Trans 29 (1):1-3. doi:10.1016/0019-0578(90)90023-e
8. Jordan MI, Mitchell TM (2015) Machine learning: Trends, perspectives, and prospects. Science 349 (6245):255-260. doi:10.1126/science.aaa8415
9. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85-117. doi:10.1016/j.neunet.2014.09.003
10. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521 (7553):436-444. doi:10.1038/nature14539
11. Yang X, Wang Y, Byrne R, Schneider G, Yang S (2019) Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery. Chem Rev 119 (18):10520-10594. doi:10.1021/acs.chemrev.8b00728
12. Polishchuk PG, Madzhidov TI, Varnek A (2013) Estimation of the size of drug-like chemical space based on GDB-17 data. J Comput Aided Mol Des 27 (8):675-679. doi:10.1007/s10822-013-9672-4
13. Hughes JP, Rees S, Kalindjian SB, Philpott KL (2011) Principles of early drug discovery. Br J Pharmacol 162 (6):1239-1249. doi:10.1111/j.1476-5381.2010.01127.x
14. Vamathevan J, Clark D, Czodrowski P, Dunham I, Ferran E, Lee G, Li B, Madabhushi A, Shah P, Spitzer M, Zhao S (2019) Applications of machine learning in drug discovery and development. Nat Rev Drug Discov 18 (6):463-477. doi:10.1038/s41573-019-0024-5
15. Ashburn TT, Thor KB (2004) Drug repositioning: identifying and developing new uses for existing drugs. Nat Rev Drug Discov 3 (8):673-683. doi:10.1038/nrd1468
16. Katsila T, Spyroulias GA, Patrinos GP, Matsoukas MT (2016) Computational approaches in target identification and drug discovery. Comput Struct Biotechnol J 14:177-184. doi:10.1016/j.csbj.2016.04.004
17. Bravo A, Pinero J, Queralt-Rosinach N, Rautschka M, Furlong LI (2015) Extraction of relations between genes and diseases from text and large-scale data analysis: implications for translational research. BMC Bioinformatics 16:55. doi:10.1186/s12859-015-0472-9
18. Kim J, Kim JJ, Lee H (2017) An analysis of disease-gene relationship from Medline abstracts by DigSee. Sci Rep 7:40154. doi:10.1038/srep40154
19. Santos R, Ursu O, Gaulton A, Bento AP, Donadi RS, Bologa CG, Karlsson A, Al-Lazikani B, Hersey

A, Oprea TI, Overington JP (2017) A comprehensive map of molecular drug targets. Nat Rev Drug Discov 16 (1):19-34. doi:10.1038/nrd.2016.230

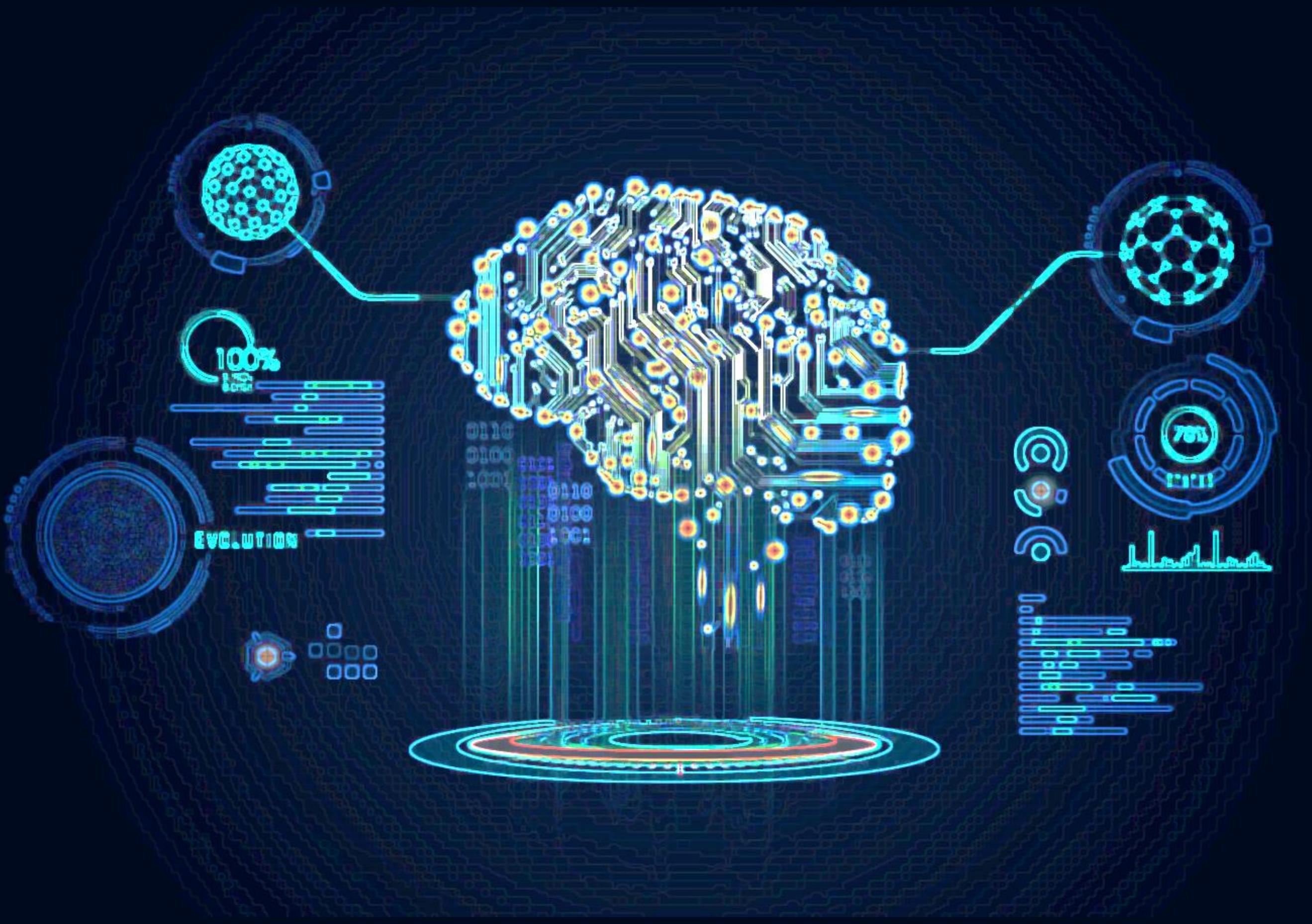
20. Thomas JM (2012) Centenary: The birth of X-ray crystallography. Nature 491 (7423):186-187. doi:10.1038/491186a
21. Pellecchia M, Sem DS, Wuthrich K (2002) NMR in drug discovery. Nat Rev Drug Discov 1 (3):211-219. doi:10.1038/nrd748
22. Renaud JP, Chari A, Ciferri C, Liu WT, Remigy HW, Stark H, Wiesmann C (2018) Cryo-EM in drug discovery: achievements, limitations and prospects. Nat Rev Drug Discov 17 (7):471-492. doi:10.1038/nrd.2018.77
23. Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Zidek A, Potapenko A, Bridgland A, Meyer C, Kohl SAA, Ballard AJ, Cowie A, Romera-Paredes B, Nikolov S, Jain R, Adler J, Back T, Petersen S, Reiman D, Clancy E, Zielinski M, Steinegger M, Pacholska M, Berghammer T, Bodenstein S, Silver D, Vinyals O, Senior AW, Kavukcuoglu K, Kohli P, Hassabis D (2021) Highly accurate protein structure prediction with AlphaFold. Nature. doi:10.1038/s41586-021-03819-2
24. Fox S, Farr-Jones S, Sopchak L, Boggs A, Nicely HW, Khoury R, Biros M (2006) High-throughput screening: update on practices and success. J Biomol Screen 11 (7):864-869. doi:10.1177/1087057106292473
25. van Westen GJ, Hendriks A, Wegner JK, IJzerman AP, van Vlijmen HW, Bender A (2013) Significantly improved HIV inhibitor efficacy prediction employing proteochemometric models generated from antivirogram data. PLoS Comput Biol 9 (2):e1002899. doi:10.1371/journal.pcbi.1002899
26. van Westen GJ, Wegner JK, Geluykens P, Kwanten L, Vereycken I, Peeters A, IJzerman AP, van Vlijmen HW, Bender A (2011) Which compound to select in lead optimization? Prospectively validated proteochemometric models guide preclinical development. PLoS One 6 (11):e27518. doi:10.1371/journal.pone.0027518
27. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T (2018) The rise of deep learning in drug discovery. Drug discovery today 23 (6):1241-1250. doi:10.1016/j.drudis.2018.01.039
28. Kitchen DB, Decornez H, Furr JR, Bajorath J (2004) Docking and scoring in virtual screening for drug discovery: methods and applications. Nat Rev Drug Discov 3 (11):935-949. doi:10.1038/nrd1549
29. Hollingsworth SA, Dror RO (2018) Molecular Dynamics Simulation for All. Neuron 99 (6):1129-1143. doi:10.1016/j.neuron.2018.08.011
30. Mullard A (2018) Re-assessing the rule of 5, two decades on. Nat Rev Drug Discov 17 (11):777. doi:10.1038/nrd.2018.197
31. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, Light Y, McGlinchey S, Michalovich D, Al-Lazikani B, Overington JP (2012) ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic Acids Res 40 (Database issue):D1100-1107. doi:10.1093/nar/gkr777
32. Irwin JJ, Shoichet BK (2005) ZINC--a free database of commercially available compounds for virtual screening. Journal of chemical information and modeling 45 (1):177-182. doi:10.1021/ci049714+
33. Cincilla G, Thormann M, Pons M (2010) Structuring Chemical Space: Similarity-Based Characterization of the PubChem Database. Mol Inform 29 (1-2):37-49. doi:10.1002/minf.200900015
34. Segler MHS, Preuss M, Waller MP (2018) Planning chemical syntheses with deep neural networks and symbolic AI. Nature 555 (7698):604-610. doi:10.1038/nature25978

35. Ahmad I, Huang L, Hao H, Sanders P, Yuan Z (2016) Application of PK/PD Modeling in Veterinary Field: Dose Optimization and Drug Resistance Prediction. Biomed Res Int 2016:5465678. doi:10.1155/2016/5465678
36. Sanguinetti MC, Tristani-Firouzi M (2006) hERG potassium channels and cardiac arrhythmia. Nature 440 (7083):463-469. doi:10.1038/nature04710
37. Milnes JT, Crociani O, Arcangeli A, Hancox JC, Witchel HJ (2003) Blockade of HERG potassium currents by fluvoxamine: incomplete attenuation by S6 mutations at F656 or Y652. Br J Pharmacol 139 (5):887-898. doi:10.1038/sj.bjp.0705335
38. van Gool AJ, Bietrix F, Caldenhoven E, Zatloukal K, Scherer A, Litton JE, Meijer G, Blomberg N, Smith A, Mons B, Heringa J, Koot WJ, Smit MJ, Hajduch M, Rijnders T, Ussi A (2017) Bridging the translational innovation gap through good biomarker practice. Nat Rev Drug Discov 16 (9):587-588. doi:10.1038/nrd.2017.72
39. Kraus VB (2018) Biomarkers as drug development tools: discovery, validation, qualification and use. Nat Rev Rheumatol 14 (6):354-362. doi:10.1038/s41584-018-0005-9
40. Min S, Lee B, Yoon S (2017) Deep learning in bioinformatics. Brief Bioinform 18 (5):851-869. doi:10.1093/bib/bbw068
41. Murdoch WJ, Singh C, Kumbier K, Abbasi-Asl R, Yu B (2019) Definitions, methods, and applications in interpretable machine learning. Proc Natl Acad Sci U S A 116 (44):22071-22080. doi:10.1073/pnas.1900654116
42. Angermueller C, Parnamaa T, Parts L, Stegle O (2016) Deep learning for computational biology. Mol Syst Biol 12 (7):878. doi:10.15252/msb.20156651
43. Finnegan A, Song JS (2017) Maximum entropy methods for extracting the learned features of deep neural networks. PLoS Comput Biol 13 (10):e1005836. doi:10.1371/journal.pcbi.1005836
44. Hutson M (2018) Artificial intelligence faces reproducibility crisis. Science 359 (6377):725-726. doi:10.1126/science.359.6377.725
45. Lv X, Liu J, Shi Q, Tan Q, Wu D, Skinner JJ, Walker AL, Zhao L, Gu X, Chen N, Xue L, Si P, Zhang L, Wang Z, Katritch V, Liu ZJ, Stevens RC (2016) In vitro expression and analysis of the 826 human G protein-coupled receptors. Protein & cell 7 (5):325-337. doi:10.1007/s13238-016-0263-8
46. Trzaskowski B, Latek D, Yuan S, Ghoshdastider U, Debinski A, Filipek S (2012) Action of molecular switches in GPCRs--theoretical and experimental studies. Curr Med Chem 19 (8):1090-1109. doi:10.2174/092986712799320556
47. Dorsam RT, Gutkind JS (2007) G-protein-coupled receptors and cancer. Nature reviews Cancer 7 (2):79-94. doi:10.1038/nrc2069
48. Hauser AS, Attwood MM, Rask-Andersen M, Schioth HB, Gloriam DE (2017) Trends in GPCR drug discovery: new agents, targets and indications. Nature reviews Drug discovery. doi:10.1038/nrd.2017.178
49. Jacobson KA, Gao ZG (2006) Adenosine receptors as therapeutic targets. Nat Rev Drug Discov 5 (3):247-264. doi:10.1038/nrd1983
50. Chen JF, Eltzschig HK, Fredholm BB (2013) Adenosine receptors as drug targets--what are the challenges? Nat Rev Drug Discov 12 (4):265-286. doi:10.1038/nrd3955
51. Delacretaz E (2006) Clinical practice. Supraventricular tachycardia. N Engl J Med 354 (10):1039-1051. doi:10.1056/NEJMcp051145
52. Moro S, Gao ZG, Jacobson KA, Spalluto G (2006) Progress in the pursuit of therapeutic adenosine receptor antagonists. Med Res Rev 26 (2):131-159. doi:10.1002/med.20048

53. Saaksjarvi K, Knekt P, Rissanen H, Laaksonen MA, Reunanen A, Mannisto S (2008) Prospective study of coffee consumption and risk of Parkinson's disease. Eur J Clin Nutr 62 (7):908-915. doi:10.1038/sj.ejcn.1602788

Chapter 2

Computational approaches for *de novo* drug design: past, present and fut**ure**



Xuhan Liu. Adriaan P. IJzerman and Gerard J. P. van Westen*. Methods in Molecular Biology (2021). Https://doi.org/10.1007/978-1-0716-0826-5_6

Abstract

Drug discovery is time- and resource-consuming process. To this end, computational approaches that are applied in *de novo* drug design play an important role to improve the efficiency and decrease costs to develop novel drugs. Over several decades, a variety of methods have been proposed and applied in practice. Traditionally, drug design problems are always taken as combinational optimization in discrete chemical space. Hence optimization methods were exploited to search for new drug molecules to meet multiple objectives. With the accumulation of data and the development of machine learning methods, computational drug design methods have gradually shifted to a new paradigm. There has been particular interest in the potential application of deep learning methods to drug design. In this chapter, we will give a brief description of these two different *de novo* methods, compare their application scopes and discuss their possible development in the future.

Keywords: machine learning, cheminformatics, deep learning, drug discovery, optimization

2.1. Introduction

Drug discovery is always considered to have a significant "serendipity" component, --researchers need to identify a small fraction of feasible molecules with desired physicochemical and biological properties from the vast chemical space, which has been estimated to be comprised of 10^{23}~10^{60} feasible drug-like molecules [1]. This number of potential candidate molecules is too large to screen experimentally. Moreover, drug molecules have a high promiscuity [3], *i.e.* each drug-like molecule has six protein targets on average, leading to the unexpected toxicity and withdrawal of some FDA approved drugs from the market [4]. These problems have contributed to an increase in the average cost to over one billion USD for the development of a new drug in a process that takes about 13 years to reach the market [5].

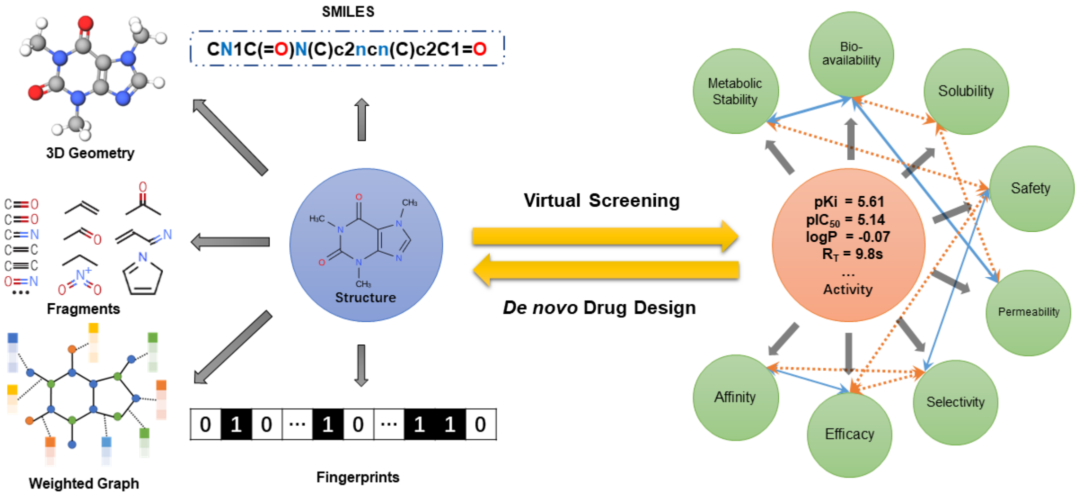


Fig. 2.1: Schematic overview of the interplay of two methods in computational drug discovery: virtual screening and *de novo* design. The left of the figure shows ways in which a molecule can be described for computational methods (see 'Molecular Representations'). On the right the multi-objective nature of the problem is shown. Properties are often contrary (orange arrows) and sometimes cooperative (blue arrows), but must be optimized simultaneously (see 'Multiple Objectives').

To this end, computer-aided drug discovery (CADD) aims to speed up the drug discovery process by integrating chemical and biological information about ligands and/or targets [6]. CADD is a broad field of research that includes *de novo* drug design and virtual screening methods (Fig. 2.1, center). *De novo* drug design suggests new molecules as starting points for chemical modifications that result in novel leads. By contrast, virtual screening methods try to uncover the hidden relationships between chemical structure and pharmacological activity. CADD has always been a combinatorial optimization problem with multi-objective optimization. Virtual screening methods provide a scoring function that mimics bioassays in order to guide the drug design algorithm to converge on the optimal molecule. Because it is impossible to enumerate every chemical entity in the chemical universe, CADD in practice does not lead to a globally optimal solution, but it narrows down the searching scope of chemical space and converges on a local or practical optimum [7].

In the past, machine learning methods, such as random forests, were mainly constructed for virtual screening, i.e. given the structure of a chemical compound predict its biological activity. With the increased availability of (public) data and development of computer sciences (e.g. the introduction of GPU computation), machine learning methods have also found their way to the field of *de novo* drug design. Deep learning (DL) methods in particular have attracted increasing attention as a promising approach for drug discovery [8]. DL methods are an extension of artificial neural networks that add a variety of multiple hidden layers, thus making the network significantly deeper [9]. In 2012, deep convolutional neural networks (CNNs) were proposed and became a breakthrough in image classification [10]. Subsequently, generative adversarial networks (GANs) were developed for image generation and, by 2014, these had significantly improved the quality of generated images [11]. Based on these achievements, the DL methods could also provide a series of solutions for prediction, generation, and decision-making in other data rich fields beyond image recognition and natural language processing [8]. In drug discovery, DL has catalyzed an explosion of applications for *de novo* drug design since Gómez-Bombarelli *et al.* applied variational autoencoders (VAE) to generate SMILES-based chemical compounds in 2016 [13].

As traditional optimization algorithms and recent DL methods are quite distinct, it is necessary to make a clear comparison between both methods. In the following paragraphs, we will give more theoretical details of these two different methods and their application in the field of drug design. We will also discuss the advantages and disadvantages of both of them and possible directions of their combination in the future.

2.2. *De novo* drug design

Due to the discreteness of chemical space, drug design is intuitively rendered into a combinatorial optimization problem. The solution of this drug design problem is searching for an optimal combination of building blocks to find the best solution according to the required conditions. Based on the difference of the building blocks, drug design algorithms can be classified into atom-based and fragment-based methods. The atom-based methods are the more intuitive approaches and easily construct a variety of novel structures, but are more time-consuming and less able to converge to the best solutions. In contrast, fragment-based methods reduce the chemical space dramatically by pre-defining the fragment library and are consequently faster searching for optimal molecules than atom-based methods, although the diversity is lower compared to atom-based methods. However, the drug design problem cannot be solved completely, because an increase in fragments leads to a combinatorial explosion of chemical space, making an exhaustive search impossible. Therefore, more efficient molecular representations need to be developed to suggest novel potential drug-like molecules efficiently in addition to, or as an alternative for the known atomistic and fragment-based representations.

Usually, drug molecules are organic compounds with physiochemical properties optimal for drug-like molecules, such as Lipinski’s rule of 5. Moreover, sufficient on-target affinity and avoiding off-target affinity are additional objectives that need to be met.

Drug *de novo* design can be further classified into structure-based and ligand-based methods based on whether 3D structure information is available and included [7,14]. In structure-based drug design, the 3D structure of a protein target is required for guiding ligand design but prior knowledge of other ligands is unnecessary. The optimal ligands are commonly obtained by calculating the binding energy when combining at the protein active site to interact with the protein. This compares with ligand-based methods, which do not exploit protein target structure information but require the prior knowledge comprised of known ligands of given structures which are used to measure their similarity with generated molecules.

2.2.1. Molecular representations

Chemical compounds are not a random cluster of atoms and functional groups, but rather have a definite structure represented by the arrangement of chemical bonds between atoms and information on the geometric 3D shape. This information needs to be represented computationally for algorithms to be able to predict properties of these molecules (Fig. 2.1). Ideally, the full 3D shape geometry is used for construction of a fitness function in structure-based optimization methods, such as docking or molecular dynamics [15]. However, these 3D approaches always consume more computational resources and time; they also require the computational generation of conformers, a process which can be prone to error.

To circumvent this requirement 2D approaches are used. As the key to properties of the molecules lies in fragments with a specific connection pattern of the atoms, molecules can be represented as a bag of fragments which can be perturbated easily for generating new molecules (in the form of a binary bit string). This molecular fingerprint can also be used as input for virtual screening [16]. A downside to fingerprints is that the connectivity information linking the individual fragments is not available. Hence various different molecules can be generated with the same combination of fragments. Moreover, while each fragment of the molecule can be mapped to one bit in a fingerprint by a hash function, such as ECFP [17], the fingerprint is always irreversible. A fingerprint cannot be reconstructed into a molecule, so it is impossible to use the molecular fingerprint directly for drug design. All in all, there is no single 2D or 3D representation that seems to meet all criteria [18].

To circumvent the loss of connectivity information, other methods are used. The most natural molecular representation is an undirected graph where the atoms and bonds are nodes and edges respectively [19]. These graphs can be reversibly converted into a text format using a preset grammar such as simplified molecular-input line-entry specification (SMILES). Analogous to natural language processing, SMILES is regarded as a chemical language and directly used in deep learning models for molecular generation. However, as SMILES follows a fixed grammar, generated texts can easily lead to invalid molecules. To solve this problem, some groups attempted to decompose SMILES into a sequence of rules from a context free grammar and improved linear molecular representation, such as DeepSMILES [20], Randomized SMILES [21], and SELIES [22]. An advanced representation is directly storing the graph into multi-dimensional tensors, including type of atoms and edges, and connectivity information. This representation can make sure the molecular graph can be generated immediately without considering grammar; however, it is still computationally expensive.

2.2.2. Multiple objectives

As specified above, drug design is always a multi-objective problem (MOP) and designed compounds need to meet many criteria as drug candidates *e.g.* efficacy, selectivity, safety, permeability, solubility, metabolic stability, synthesizability, *etc*. (Fig. 2.1) Some of these objectives are not independent but contradictory, meaning that if an optimum is achieved on one objective it has been at the expense of making a compromise on other objectives. Unlike single-objective problems (SOP), where the best solution is on the top of ranking sorted by the scalar score of each candidate solution, the ranking of candidates in a MOP is more complicated because of conflicting objectives [14]. A straightforward method of dealing with this complication is to convert the multiple objectives into a single objective by weighted summing of scores for each objective [23].

$$f(n) = \sum_{i=1}^{N} w_i p_i$$

where $f(n)$ is the fitness function and w_i is pre-defined by users as the weight of i^{th} objective p_i. However, it is challenging to determine these weights, because they specify a single pattern of compromise for these objectives, which can trap an optimization algorithm and lead to unreasonable solutions.

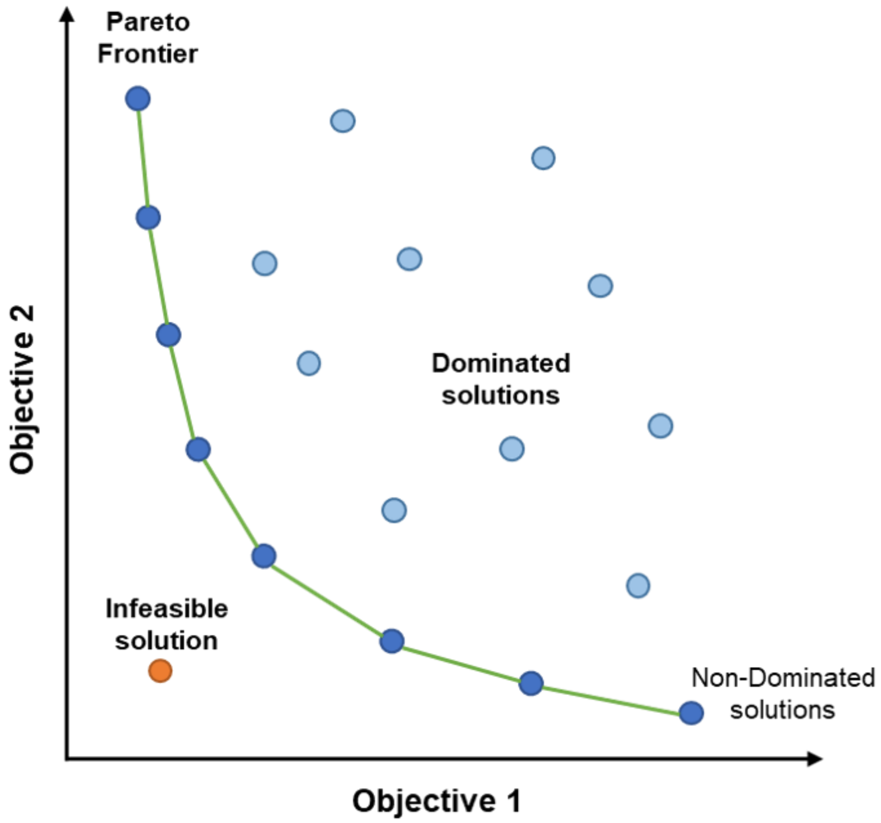


Fig. 2.2: Pareto frontier in multi-objective optimization. Take two objectives as an example, non-dominated solutions form a boundary called Pareto frontier which separates the infeasible solutions in the lower left region from dominated solutions in the upper right region.

In order to strike a better balance between each objective, MOP algorithms produce a set of solutions representing various compromises among the objectives. The solutions are mapped out on a hypersurface in the search space, termed Pareto Front [24]. A solution dominates another one if it is equivalent or better in all objectives and better in at least one objective compared with all other solutions. Solutions with the most appropriate compromise among the individual objectives can be identified through pareto ranking. Several pareto ranking algorithms have been developed (*e.g.* SPEA [25], NSGA [26], SMS-EMOA [27], *etc*). However, all of them are computationally expensive for large numbers of objectives and data points and lead to non-convergence of the solutions in contradiction of the SOP [23].

2.3. Optimization methods

In applications of drug design, the most popular searching algorithms are evolutionary algorithms (EAs), particle swarm optimization (PSO), and Simulated annealing (SA) (Table 2.1). In the following paragraphs, we will briefly introduce their mathematical theories and their application in drug discovery.

2.3.1. Evolutionary algorithms

EAs are population-based metaheuristic optimization algorithms inspired by biological evolution to mimic the genetic operators, such as “reproduction”, “mutation”, and “crossover” [44]. In the population, a pair of individuals is randomly selected for each time and play the role of parents to “reproduce” the offspring through “mutation” and “crossover” for population expansion. The scoring function, also called a fitness function in EAs, determines which individual can survive and replace the least-fit individual in the population. The surviving individuals in the updated population are selected as the new parents for next generation. For each iteration of the evolutionary cycle, the average fitness score of individuals in the population will be improved and this cyclic process will continue until a termination criterion is reached. Currently, EAs are the most sophisticated algorithm used for drug *de novo* design in practice.

There are several major algorithmic techniques in use in EAs, examples include genetic algorithms, genetic programming, and evolutionary strategies [45]. Genetic algorithms (GAs) are one of the most fundamental and widely used EAs. GAs need to encode the phenotype (molecular structure) by means of a ‘chromosome’ as the simulation of natural selection [46]. For example, Wang *et al.* developed a software named LigBuilder, in which each molecule was decomposed into a series of fragments from the building-block library to be used as ‘chromosome’ [28]. The mutation operator was defined to allow only carbon, nitrogen, and oxygen atoms of the molecules with the same hybridization state to mutate to each other. During the process, fragments were combined to generate a new population through randomly selecting a growing site on the seed structure and addition of a fragment from the building-block library. Each molecule was represented with its SMILES sequence as the ‘chromosome’. Similarly, Douguet *et al.* defined allowable crossover points and mutation rules were generated for breeding valid SMILES as the next generation in their method deemed LEA [29].

Table 2.1: Current optimization methods for *de novo* drug design.

Methods	Method	Molecule Representation	Objective	Reference
LigBuilder	GA	3D geometry	Affinity (Thrombin and dihydrofolate reductase) and Bioavailability Score	Wang et al. [28]
LEA	GA	SMILES	Analogs fitness (Retinoid and Salicylic Acid) and physico-chemical properties	Douguet et al. [29]
ADAPT	GA	Fragment	Docking score (cathepsin D, dihydrofolate reductase, and HIV-1 reverse transcriptase), RO5	Pegg et al. [30]
PEP	GA	Fragment	Force field-based binding energy (Caspase 1, 3 and 8)	Budin et al. [31]
SYNOPSIS	GA, SA	Reactivity	Electric dipole moment, affinity to binding site (HIV-1 reverse transcriptase)	Vinkers et al. [32]
LEA3D	GA	Fragment	Molecular Properties, Affinity to binding site (thymidine monophosphate kinase)	Douguet et al. [33]
GANDI	GA	Fragment	2D/3D similarities and force field-based binding energy (cyclin-dependent kinase 2)	Dey et al. [34]
Molecule Commander	GA	Fragment	Affinity to A_1AR, off-target selectivity (A_{2A}AR A_{2B}AR A_3AR) and ADMET scores	van der Horst et al. [35]
Molecule Evoluator	GP	Tree SMILES	QSAR functions, docking, experiments, similarity to template molecules (Neuramidase inhibitor)	Lameijer et al. [36]
MEGA	GP	Graph	Binding affinity score (Estrogen receptor), similarity score and RO5	Nicolaou et al. [37]
FLUX	ES	Fragment	Similarity to template molecules (tyrosine kinase inhibitor, Factor Xa inhibitor)	Fechner et al. [38]
TOPAS	ES	Fragment	2D structural/topological pharmacophore similarity to template (thrombin inhibitor)	Schneider et al. [39]
MOLig	SA	Fragment	Force field-based binding energy (RecA), similarity to template molecules, oral bioavailability	Sengupta et al. [40]
CONCERTS	SA	Fragment	Force field-based binding energy (FK506 binding protein, HIV-1 aspartyl protease)	Pearlman et al. [41]
SkelGen	SA	Fragment	Binding affinity prediction score (DNA gyrase and estrogen receptor)	Dean et al. [42]
COLIBREE	PSO	Fragment	Similarity to template molecules (PPAR ligands)	Hartenfeller et al. [43]

In GAs, there are fixed data structures (despite the linearity of the chromosome) to organize the variables which need to be optimized. But if these variables are interdependent through an explicit relationship, such as procedural or functional representation, genetic programming (GP) is a more suitable method to realize the EA principles [47]. In GP, the chromosomes are always represented as trees rather than the fixed-length strings of GAs. And crossover is implemented as recombination of subtrees between two parents, while mutation selects and alters a random node or edge of the tree depending on its type. Usage of a SMILES representation as a “chromosome” is troublesome for genetic operators, because SMILES *per se* is a grammatic constraint linear string and the random mutation and crossover will produce a large number of invalid SMILES. Lameijer *et al.* solved this problem in their software, named ‘Molecule Evoluator’ based on a SMILES representation employing a GP [36]. In Molecule Evoluator, TreeSMILES are defined as the tree structure being transformed from the SMILES according to its grammar, in which each node and edge denoted the atom and bond respectively. Every node or edge has an operator function, making mathematical expressions easy to evolve and evaluate.

Evolutionary strategies (ES) are a third EA technique using the concepts of adaptation and evolution. In contrast to GAs, selection in ES is based on a fitness ranking rather than fitness values, although mutation and selection also play an important role for breeding [48]. ES operates on the parent and the result of its mutants. In ES, a number of mutants are generated which compete with the parent, wherein the best mutant becomes the parent of the next generation. For example, Flux implemented a simplistic (1, λ)-ES without adaptive step-size control and defined the crossover and mutation generators on the fragment-based “reaction tree” of each pair of parents [38]. Selection was performed only among the offspring and the parent died out, which could facilitate escaping local optima in the fitness landscape. Another method, TOPAS, used a simple (1, λ)-ES with adaptive parameters [39]. During the stochastic search process, there were λ=100 variants generated through virtual synthesis for each iteration. The distribution of Tanimoto similarity with their parents was controlled by a step-size parameter, which guaranteed that the chemical space of the population adapts to the local shape of the fitness landscape. Similarly, only one variant with the best fitness score became the parent of the next generation while the current parent was discarded.

2.3.2. Particle swarm optimization

PSO solves the optimization problem based on the observation of collective intelligence in many natural systems that individuals cooperate with each other to improve not only their collective performance but also each individual's performance on a given task [49]. Similar to EAs, PSO also is a population-based method. In PSO a population, known as a swarm, contains a series of candidate solutions (called particles). The population needs to be initialized to represent the position in the search space, and the individuals should have initial velocities. In addition, each particle has its own memory to record the best fitness of its past for communication with others. In each iteration, the fitness score of each individual's position is calculated to register the best position. Subsequently the velocity of each particle is randomly influenced by two factors: one is the best-known position of a particle in its neighborhood and the other is the best position it ever searched in the past. Subsequently, the new position of each particle will be calculated based on its updated velocity. If each particle can communicate with all the other particles and share the same best position from a single particle the swarm will be trapped in a local minimum. Therefore, one of the key points is how to define the topology of the swarm to determine its neighbors.

The PSO algorithm was frequently used in continuous search spaces. In order to be applied in the discrete search space of drug-like molecules, Hartenfeller *et al.* replaced the concept of velocity of each particle with the quality vector and developed COLIBREE for drug design [43]. In COLIBREE, each molecule was represented as building blocks and linkers. The fitness function is defined as the similarity between reference ligands and generated molecules under chemically advanced template search (CATS) descriptors. Each particle stores the current search point (a molecule) and a quality vector which represented a relative probability for every fragment in the library to be chosen in the next search step for constructing the molecule. During the optimization cycle, each particle created a new molecule and updated its memory after the fitness was evaluated. The quality vector was incremented if the fragment had been part of the molecule stored in the memory of the current particle. In the end, good solutions have a higher probability to be chosen for molecule construction in subsequent search steps.

2.3.3. Simulated annealing

For the purpose of estimating a global optimum of an objective function, Simulated Annealing (SA) is based on the cooling and crystallizing behavior of chemical substances. This behavior is affected by both the temperature and the thermodynamic free energy. In general, SA sets the initial temperature and choses a random point as the initial solution. It then works iteratively in steps during which the temperature is progressively decreased from an initial value to zero. For each iteration, a new point is randomly selected from the points close to the current one as the solution. Subsequently, a probability score is calculated based on whether the quality of the new solution is better than the current solution or not, and the algorithm decides which solution will be adopted to replace the current solution. This probability is affected by the temperature, *i.e.* the temperature controls the balance of exploration/exploitation strategies. If the initial temperature is too low or cooling is too fast, the algorithm will not effectively explore the search space. Conversely, when the temperature is set too high, the algorithm will take too long to converge. The key point of SA is the strategy about how to choose a new solution, which has a significant impact on its performance.

Sengupta *et al.* developed MOLig with the SA algorithm in 2012 [40]. This method encoded each molecule into a tree-like representation which was stored as an array of positive integers. In this array numbers symbolized a molecular fragment and specified the connectivity pattern. For each iteration, there were several perturbation operators being defined for generating molecules as a new solution and it would be determined by temperature related probability whether this new solution would replace the current one. The iteration would terminate once the temperature was reduced to zero. In addition, CONCERTS [41] and SkelGen [42] are other structure-based *de novo* design methods based on the SA algorithm.

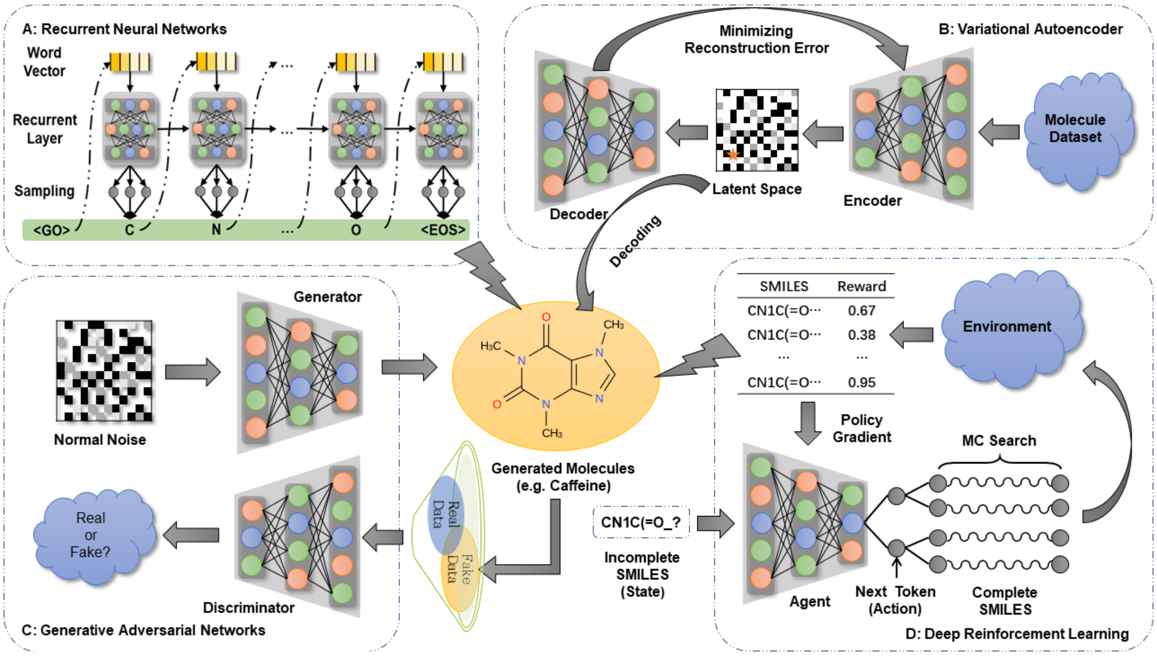


Fig. 2.3: Four basic deep learning architectures commonly used in de novo drug design, including recurrent neural networks (A), variational autoencoder (B), generative adversarial networks (C) and deep reinforcement learning (D).

2.4. Deep learning algorithms

The common basic DL architectures used in *de novo* drug design are recurrent neural networks (RNNs), variational autoencoder (VAE), deep reinforcement learning (RL), and generative adversarial networks (GANs) (Fig. 2.3). Most studies of DL applications combine two or more models to address specific issues. In the following paragraphs, we give the details about these architectures, and how these models can be applied in drug design. We also list and categorize these methods based on these DL architectures in Table 2.2.

2.4.1. Recurrent neural networks

RNNs can process sequential data effectively because the connections between neurons form a directed acyclic graph that can be unrolled along the temporal sequences [81]. RNNs have shown excellent performance in the field of natural language processing (NLP) such as handwriting [82] or speech recognition [83]. RNNs deal with words in text step by step and deliver the current hidden information to the next step in the network with the same structure simultaneously. By analogy, the direct application of RNNs in drug design takes the linear molecular representations as input [61,60,53]. For example, SMILES are always preprocessed by being split into a sequence of tokens $\boldsymbol{x_{1:n}} = [\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}]$. The SMILES string is then prefixed with a start token $\boldsymbol{x_0}$ as input feature and suffixed with the end token $\boldsymbol{x_{n+1}}$ as the output labels. The RNN model $\boldsymbol{\pi_\theta}$ parametrized by $\boldsymbol{\theta}$ determines the probability distribution $\boldsymbol{y_i}$ of tokens based on $\boldsymbol{x_{0:i-1}}$:

$$\boldsymbol{h_i} = \boldsymbol{f}_r(\boldsymbol{h_{i-1}}, \boldsymbol{x_{i-1}})$$

$$\boldsymbol{y_i} = \boldsymbol{f_o}(\boldsymbol{h_i})$$

here, $\boldsymbol{f_r}$ denotes recurrent layers and receives the last hidden states $\boldsymbol{h_{i-1}}$ and input features $\boldsymbol{x_{i-1}}$ to calculate the current hidden states $\boldsymbol{h_i}$. In order to avert the problem of long-distance dependencies caused by gradients vanishing or exploding, many variational versions have been proposed, including two common implementations: long short-term memory (LSTM) [84] and gated recurrent unit (GRU) [85], which contain a memory cell and some different gates to determine forgotten and reserved information. In the end, $\boldsymbol{h_i}$ are delivered to output layers $\boldsymbol{f_o}$ for calculation of output values $\boldsymbol{y_i}$ and commonly, the probability of each word in the vocabulary is computed by the SoftMax function. For the model training, the maximum likelihood estimation (MLE) is always chosen to calculate the loss function:

$$\mathcal{L}_{MLE} = \sum_{j=1}^{m} \sum_{i=1}^{n+1} \log \boldsymbol{\pi_\theta}(\boldsymbol{x_i} | \boldsymbol{x_{0:i-1}})$$

here, *m* is the total number of samples with sequence length *n* in the training set. The MLE loss function can be optimized with the backpropagation algorithm commonly used for DL model training.

The RNN model always serves as one of the basic components in the more complicated DL architectures, which will be introduced in the following paragraphs. If used independently, RNN models are often beneficial for molecular library generation. For example, Segler *et al.* pre-trained an RNN model on the ChEMBL database containing 1.4 million molecules and employed ‘transfer learning’, also called ‘fine-tuning’ methods to make molecules focused on the chemical space for the 5-HT$_{2A}$ receptor [86]. To improve the efficiency of desired molecular generation, Yang *et al*. proposed a method they termed *ChemTS* by combining an RNN model with Monte Carlo tree search [53]. Subsequently this method was successfully applied and several molecules were synthesized and confirmed to be desirable chemical compounds [87]. To balance validity and diversity of molecular generation, Gupta *et al.* modified the SoftMax function as follows:

$$P_k = \frac{\exp\left(y_k/T\right)}{\sum_k \exp\left(y_k/T\right)}$$

by adding a temperature factor *T* to rescale the probability of each token *k* in the vocabulary [61]. If temperature is increased, the diversity of molecular generation will improve, but the validation rate will decrease. Arús‑Pous et al. studied the performance of an RNN model for molecular generation on the GPB-13 dataset and found that it always fails to generate complex molecules with many rings and heteroatoms due to the syntax of SMILES [88].

Table 2.2: The current DL-based *de novo* drug design methods

Methods	**Molecular Representations**	**Architectures**	**Database**	**Objectives**	**References**
LatentGAN	SMILES	VAE, GAN	ChEMBL, ExCAPE-DB	Affinity to EGFR, HTR1A and S1PR1	Oleksii, et al.[50]
ANTC	SMILES	DNC, GAN, RL	ChemDiv	Similarity, Diversity, QED and presence of sp3-rich fragments	Putin et al.[51]
	SMILES	AAE	ChEMBL, ExCAPE-DB	Affinity to DRD2	Blaschke et al. [52]
ChemVAE	SMILES	VAE	QM9, ZINC	SAS and QED	Gómez-Bombarelli et al. [13]
ChemTS	SMILES	RNN, MCTS	ZINC	logP SAS and ring penalty	Yang et al. [53]
SSVAE	SMILES	VAE	ZINC	Drug-likeness	Kang et al. [54]
		VAE, BO	ZINC	logP, SAS, QED and ring penalty	Griffiths et al. [55]
	SMILES	RNN, TL	ChEMBL	Affinity to PPAR and RXR	Merk et al. [56]
	SMILES	VAE, GTM	ChEMBL	Affinity to $A_{2a}R$	Sattarov et al. [57]
ReLeaSE	SMILES	RL	ZINC, ChEMBL	Affinity to JAK2	Popova et al. [58]
	SMILES	AAE	ZINC	Affinity to JAK2 and JAK3	Polykovskiy et al. [59]
	SMILES	RNN, TL	ChEMBL	Targeting the 5-HT2A receptor, Malaria and Golden Staph	Segler et al. [60]

	SMILES	RNN	ChEMBL	Affinity to PPARγ, TRPM8 and Trypsin	Gupta et al. [61]
	SMILES, Inchi	RNN, PSO	ChEMBL, SureChEMBL	logP, SAS, QED and Affinity to EGFR and BACE1	Winter et al. [62]
	SMILES	RNN	ChEMBL, GDB-8	Diversity	Bjerrum et al. [63]
	SMILES	VAE	ZINC	Drug-likeness	Lim et al. [64]
DrugEx	SMILES	RL, RNN	ZINC, ChEMBL	Diversity and Affinity to A_{2A}AR	Liu et al. [65]
REINVENT	SMILES	RL, RNN	ChEMBL	Affinity to DRD2	Olivecrona et al. [66]
MolDQN	Atoms/Bonds	RL	ChEMBL, ZINC	logP, SAS and QED	Zhou et al. [67]
ORGAN	SMILES	RNN, RL, GAN	GDB-17, ChEMBL	logP, SAS and QED	Guimaraes et al. [68]
RANC	SMILES	DNC, RL, GAN	ZINC, ChemDiv	Drug-likeness	Putin et al. [69]
SD-VAE	SMILES	VAE	ZINC	Validation of Molecule	Dai et al. [70]
GrammarVAE	SMILES	VAE, BO	ZINC	Validation of Molecule	Kusner et al. [71]
LigDream	SMILES, 3D Geometry	VAE, CNN, RNN	ZINC, DUDE	Affinity to A_{2A}AR, THRB and KIT	Skalic et al. [72]
	3D geometry	GCN	scPDB, BMOAD	Affinity to given protein	Aumentado-Armstrong [73]
GraphVAE	Graph	VAE	QM9	Validation of Molecule	Simonovsky et al. [74]
CGVAE	Graph	VAE	QM9, ZINC, CEPDB	QED	Liu et al. [75]
GCPN	Graph	GCN, RL	ZINC	logP, SAS and QED	Yu et al. [76]
JT-VAE	Graph	VAE	ZINC	logP, SAS and Ring Penalty	Jin et al. [77]
MolecularRNN	Graph	RNN, RL	ZINC	logP, SAS and QED	Popova et al. [78]
MolGAN	Graph	GAN, RL, GCN	QM9, GDB-17		De Cao et al.
MOLECULE CHEF	Graph	VAE, GGNN, RNN	USPTO	Synthesizability	Bradshaw et al. [79]
DeepFMPO	Fragment	RL	ChEMBL	Affinity to DRD2 and DRD4	Ståhl et al. [80]

2.4.2. Variational autoencoders

Variational autoencoders (VAEs) are a frequently used DL method aiming to learn representations for dimensionality reduction in an unsupervised manner [89]. The architecture of autoencoders consists of an DL-based encoder and decoder. The encoder maps the high-dimensional input data into a latent space with lower dimensional representation, whereas the decoder reconstructs these representations in the latent space into the original inputs. VAEs are a probabilistic generative model based on a directed graph with an autoencoder-like structure, while its mathematical basis, which is derived from the theory of variational inference, has little to do with traditional autoencoders [90].

The datapoint $\boldsymbol{z}$ in the latent space can be transformed into input data $\boldsymbol{x}$ by the decoder which estimates the likelihood $\boldsymbol{p_\theta}(\boldsymbol{x}|\boldsymbol{z})$ with parameters $\boldsymbol{\theta}$. In order to train the model, a straightforward approach is maximizing the distribution of input data $\boldsymbol{p}(\boldsymbol{x})$ which is approximated by $\boldsymbol{p}(\boldsymbol{x}) = \int \boldsymbol{p_\theta}(\boldsymbol{x}|\boldsymbol{z})\boldsymbol{p}(\boldsymbol{z})\, \boldsymbol{dz}$[91]. Due to the intractability of this integral, the encoder is introduced to learn a posterior $\boldsymbol{q_\varphi}(\boldsymbol{z}|\boldsymbol{x})$ parameterized by $\boldsymbol{\varphi}$; the formula for computing $\boldsymbol{p}(\boldsymbol{x})$ can be rewritten as:

$$\mathbb{E}_{q_\varphi(z|x)}[\log \boldsymbol{p}(\boldsymbol{x})] = D_{KL}\left(\boldsymbol{q_\varphi}(\boldsymbol{z}|\boldsymbol{x})||\boldsymbol{p_\theta}(\boldsymbol{z}|\boldsymbol{x})\right) + \mathbb{E}_{q_\varphi(z|x)}\left[\log \boldsymbol{p_\theta}(\boldsymbol{x},\boldsymbol{z}) - \log \boldsymbol{q_\varphi}(\boldsymbol{z}|\boldsymbol{x})\right]$$

The first term in the right hand side is Kullback-Leibler (KL) divergence and the second term is called the evidence lower bound (ELBO). Because of the non-negativity of the KL divergence, the ELBO is a lower bound of the log $\boldsymbol{p}(\boldsymbol{x})$ and is also rewritten as:

$$\mathcal{L}(\varphi,\theta) = \mathbb{E}_{q_\varphi(z|x)}\left[\log \boldsymbol{p_\theta}(\boldsymbol{x}|\boldsymbol{z})\right] - D_{KL}\left(\boldsymbol{q_\varphi}(\boldsymbol{z}|\boldsymbol{x})||\boldsymbol{p}(\boldsymbol{z})\right)$$

In order to obtain maximization of $\boldsymbol{p}(\boldsymbol{x})$, ELBO can be regarded as an objective function and maximized for training both the encoder and decoder simultaneously. Commonly in VAEs, $\boldsymbol{p}(\boldsymbol{z})$ is assumed as a unit normal Gaussian distribution and $\boldsymbol{q_\varphi}(\boldsymbol{z}|\boldsymbol{x})$ is chosen as a factorized Gaussian distribution:

$$\boldsymbol{p}(\boldsymbol{z}) \sim \boldsymbol{\mathcal{N}}(0, \mathbf{I})$$

$$\boldsymbol{q_\varphi}(\boldsymbol{z}|\boldsymbol{x}) \sim \boldsymbol{\mathcal{N}}(\mu, \mathrm{diag}(\sigma^2))$$

and the output of the encoder is shifted to output the value of the mean and the variance for the Gaussian distribution. During the training process through backpropagation, the reconstruction error of the decoder is reduced by maximizing the first term of ELBO and the encoder estimates a more accurate posterior by minimizing the KL divergence with the true *priori* of latent variables.

In 2016, Gómez-Bombarelli *et al.* proposed *ChemVAE* which made the molecules and its descriptors reversible, *i.e.* descriptors can not only be extracted in the continuous latent space by the encoder for prediction, but also be restored to the molecules by decoder for generation [13]. In addition, VAEs can also be extended for conditional generation to design molecules with desired properties [54,64]. However, with a CNN encoder and an RNN decoder, the validation rate of SMILES generated by *ChemVAE* oscillated around 75%, which was far below the performance of pure RNN models (94%-98%). To address this issue, Kusner *et al.* represented the grammar-based SMILES into parsing tree form context-free grammar. They introduced the grammar VAE (GVAE) model which directly encodes to and from the parsing tree to ensure the validation of generated SMILES [71]. Similarly, Dai *et al.* also proposed a syntax-directed variational autoencoder (SD-VAE) inspired by syntax-directed translation for syntax and semantics check [70]. In addition, Bjerrum *et al.* combined multiple different encoders to improve the diversity of generated molecules [63]

2.4.3. Deep reinforcement learning

Reinforcement learning (RL) is modeled as a Markov decision process for the interplay between an agent and an environment [92]. The goal of RL is optimizing the agent to maximize the accumulated rewards obtained from the environment by choosing effective actions. After the agent takes an action at the current step, the environment will adapt to this step by forming a new state. For the agent, a DL model can be employed to mimic the value, which predicts expected rewards of each action or each state/action pair, or policy function, which directly provides the probability of each action. For the SMILES-based drug design problem, an RNN is commonly used to model the policy function after being pre-trained with an MLE loss function. At each step $\boldsymbol{i}$, the action a_i is introduction of a token from the vocabulary chosen by the policy function based on the current state $\boldsymbol{s_i}$, which contains all the tokens generated so far $\mathbf{s}_{\boldsymbol{i}} = [\boldsymbol{a_1}, ..., \boldsymbol{a_{i-1}}]$. The accumulated rewards $\boldsymbol{G_T}$ are the simple sum of rewards over the total steps T. The aim of RL is to maximize the expected accumulated rewards:

$$J(\boldsymbol{\theta}) = \mathbb{E}[\boldsymbol{G_T}|\boldsymbol{s_0}, \boldsymbol{\theta}] = \sum_{i=1}^{T} \boldsymbol{\pi_\theta}(\boldsymbol{a_t}|\boldsymbol{s_i}) \cdot R_i$$

Usually, the end reward R_T can be obtained immediately by the environment after the generation of SMILES has completed, the intermediate reward for the action at each step is estimated by Monte Carlo (MC) search with roll-out policy,

$$\boldsymbol{R_i} = R(\boldsymbol{s_i}) = \begin{cases} \dfrac{1}{N}\displaystyle\sum_{n=1}^{N} R(\hat{\boldsymbol{s}}_T^n),\ \ \hat{\boldsymbol{s}}_T^n \in MC(\boldsymbol{s_i}), & for\ t < T \\ R(\boldsymbol{s_T}), & for\ t = T \end{cases}$$

Because of the certainty of states after the action taken by the agent, the MC search is always removed and R_i is simplified as the end reward R_T. The expected accumulated rewards have a simple form:

$$J(\theta) = \mathbb{E}\,[\boldsymbol{G_T}|\boldsymbol{s_0}, \boldsymbol{\theta}] = R_T \sum_{i=1}^{T} \boldsymbol{\pi_\theta}(\boldsymbol{a_t}|\boldsymbol{s_i})$$

With the REINFORCE algorithm [93], parameters θ in the RNN policy function can be derived as:

$$\nabla_\theta J(\boldsymbol{\theta}) = \sum_{i=1}^{T} \mathbb{E}_{\boldsymbol{a_t} \sim \boldsymbol{\pi_\theta}}[\nabla_\theta log \boldsymbol{\pi_\theta}(\boldsymbol{a_t}|\boldsymbol{s_i}) \cdot R_i]$$

Popova *et al.* developed a method *ReLeaSE* in which a stack-augmented RNN model was used as the policy function trained with the REINFORCE algorithm. It was shown to work effectively for the generation of inhibitors towards Janus protein kinase 2 (JAK2) [58].

In addition to the policy gradient to train the policy function, Zhou *et al.* proposed another method *MolDQN* based on deep Q-learning to fit the Q-value function rather than the policy function [67]. Mathematically, for a policy $\boldsymbol{\pi}$, the value of an action $\boldsymbol{a}$ on a state $\boldsymbol{s}$ can be defined as:

$$Q_{\boldsymbol{\pi}}(s, a) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{i=t}^{T} R_i\right]$$

This action-value function calculates the future rewards of taking action $\boldsymbol{a}$ on state $\boldsymbol{s}$, and subsequent actions decided by policy $\boldsymbol{\pi}$. The optimal policy is defined as:

$$\pi^* = \arg\max_a Q_{\boldsymbol{\pi}^*}(s, a)$$

and a RNN model parameterized by θ is introduced to approximate the value function

$$\mathrm{V}(s; \theta) = \max_a Q\,(s, a; \theta)$$

This approximator can be trained by minimizing the loss function of

$$\mathcal{L}(\theta) = [R(s_i) + \gamma \mathrm{V}(s_{i+1}, \theta) - Q(s_t, a_t; \theta)]^2$$

where γ is the discount factor. By comparing with other policy-based RL methods, Zhou *et al.* argued that deep Q-learning did not need any pre-trained model and performed better than the policy gradient methods.

In order to improve the stability of RL training, Olivecrona *et al.* proposed a method named "REINVENT" [66], in which a new loss function was introduced based on the Bayesian formula for RL:

$$\mathcal{L}(\theta) = \left[log\boldsymbol{P}_{\boldsymbol{Prior}}(\boldsymbol{s_T}) + \sigma R(\boldsymbol{s_T}) - log\boldsymbol{P}_{\boldsymbol{Agent}}(\boldsymbol{s_T})\right]^2$$

The authors used all molecules in the ChEMBL database to pre-train an RNN model as the *Priori*. With the parameter σ, they integrated the reward R of each SMILES into the loss function. The final *Agent* model was regarded as the *Posteriori* and trained with the policy gradient. Finally, they successfully identified a plethora of active ligands against the dopamine D2 receptor (DRD2).

Subsequently, in order to improve the diversity of generated molecules, Liu *et al.* proposed a method *DrugEx* in which the action was not only determined by the agent policy $\boldsymbol{G_\theta}$, but also by a fixed exploration policy $\boldsymbol{G_\varphi}$ which had an identical network architecture. During the training process an "exploring rate" (ε, from 0.0 to 1.0) was defined to control which policy would take actions. At each step a random number in [0.0, 1.0] was generated. If the value was smaller than ε, the $\boldsymbol{G_\varphi}$ would determine which token would be chosen, and vice versa. This method was successfully applied to the design of ligands towards the adenosine A_{2A} receptor. [65]. DrugEx was shown to better explore the chemical space for the A_{2A} receptors and produce ligands with similar physicochemical properties to known ligands which included complex ring systems that the other methods it was compared to could not produce.

2.4.4. Generative adversarial networks

GAN models were proposed as a great breakthrough method and have been extensively applied in image recognition. A GAN contains two neural networks: the generator (*G*) and the discriminator (*D*), which contest with each other under game theory [11]. *G* commits to generating fake data to the point of confusing *D* to mistake them for real samples in the training set. The discriminator on the other hand is responsible for distinguishing between the generated fake data and the real samples. During the training loop, a batch of fake data is generated by *G*, which is used subsequently for training both *G* and *D* accompanied with real data. The objective functions were originally defined as two parts for *G* and *D*, respectively:

$$\min_{G} V(G) = \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{p_z}(\boldsymbol{z})}[log(1 - D(G(\boldsymbol{z})))]$$

$$\max_{D} V(D) = \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{p_d}(\boldsymbol{x})}[logD(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{p_g}(x)}[log(1 - D(\boldsymbol{x})))]$$

here, $\boldsymbol{p_z(z)}$ is the noise distribution, $\boldsymbol{p_d(x)}$ is the data distribution in the training set and $\boldsymbol{p_g(x)}$ is the data distribution in the generated set. These two objective functions can be joined together as a minmax game in which *G* wants to minimize *V* while *D* wants to maximize it. In order to provide a strong gradient signal to obtain the global optimality, the objective function for D is rewritten as:

$$\max_{D} V(D,G) = -log(4) + 2 \cdot D_{JS}(\boldsymbol{p_d}||\boldsymbol{p_g})$$

where $D_{JS}(\boldsymbol{p_d} \| \boldsymbol{p_g})$ is the Jensen–Shannon divergence defined as follows:

$$D_{JS}(\boldsymbol{p_d}||\boldsymbol{p_g}) = \frac{1}{2}D_{KL}\left(\boldsymbol{p_d}||\frac{\boldsymbol{p_d}+\boldsymbol{p_g}}{2}\right) + \frac{1}{2}D_{KL}\left(\boldsymbol{p_g}||\frac{\boldsymbol{p_d}+\boldsymbol{p}_g}{2}\right)$$

here, the D_{KL} is the KL divergence.

To overcome several difficulties of GANs, such as mode collapse or lack of informative convergence metrics, the Wasserstein GAN (WGAN) was proposed to ensure faster and more stable training [94]. This model replaces the Jenson-Shannon divergence with the Earth-Mover distance:

$$\mathrm{W}(p,q) = \inf_{\gamma\in\Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}\|x - y\|$$

here $\Pi(p, q)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are p and q, respectively. This distance results in a more reliable gradient signal which does not vanish during the training process. Besides the above-mentioned GAN models, there are varying forms being proposed which have been collected in the GAN ZOO [95].

For drug design, a GAN model is commonly used. To ensure that the generated molecules have similar physio-chemical properties to molecules in the training set, the GAN is combined with other neural networks to construct a hybrid DL model, such as the RL model and the VAE model. The first application of GANs for drug design was proposed in 2017, named ORGAN, in which a GAN model was trained under the RL framework for multi-objective optimization [68]. ORGAN contained one RNN generator for SMILES generation and a CNN discriminator to optimize the chemical space of generated molecules. They used linear combination methods to integrate the reward function given by discriminator (R_d) and objective function (R_c) into the final rewards (R):

$$R(\boldsymbol{x}) = \lambda R_d(\boldsymbol{x}) + (1-\lambda) R_c(\boldsymbol{x})$$

here $\lambda \in [0, 1]$ is a weight hyperparameter for balancing these two rewards. ORGAN has been demonstrated to dramatically improve the percentage of generated desired druglike molecules compared to molecules in the training set based on properties, including solubility and synthesizability. In addition, there are some other groups that also exploit the GAN model to develop their methods for molecular design, such as MolGAN [96], RANC [51], and ATNC [69].

Another GAN-based hybrid model is a combination with an adversarial autoencoder (AAE) by combining multiple VAEs [97]. Instead of minimizing KL divergence to decrease the gap between the latent distribution of output by the generator and the prespecified *priori* (*e.g.* a normal distribution), AAE uses adversarial training by introducing a DL-based model as discriminator D to tell the difference between the descriptors mapped by generated molecules and molecules in the training set, respectively. The objective function of the discriminator is written as:

$$\max_D V(D) = \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{p_d}(x)}[log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x} \sim \boldsymbol{p_g}(x)}[log(1 - D(\boldsymbol{x})))]$$

and the loss function for the VAE based generator is revised as:

$$\mathcal{L}(\varphi, \theta) = V(D) - \mathbb{E}_{\boldsymbol{q_\varphi}(\boldsymbol{z}|x)}\left[\log \boldsymbol{p_\theta}(\boldsymbol{x}|\boldsymbol{z})\right]$$

Blaschke *et al.* applied the AAE model for designing active ligands towards the dopamine receptor type 2 [52]. In addition, Polykovskiy *et al.* also successfully applied this model for generating several novel inhibitors of Janus kinase 3 (JAK3) [59].

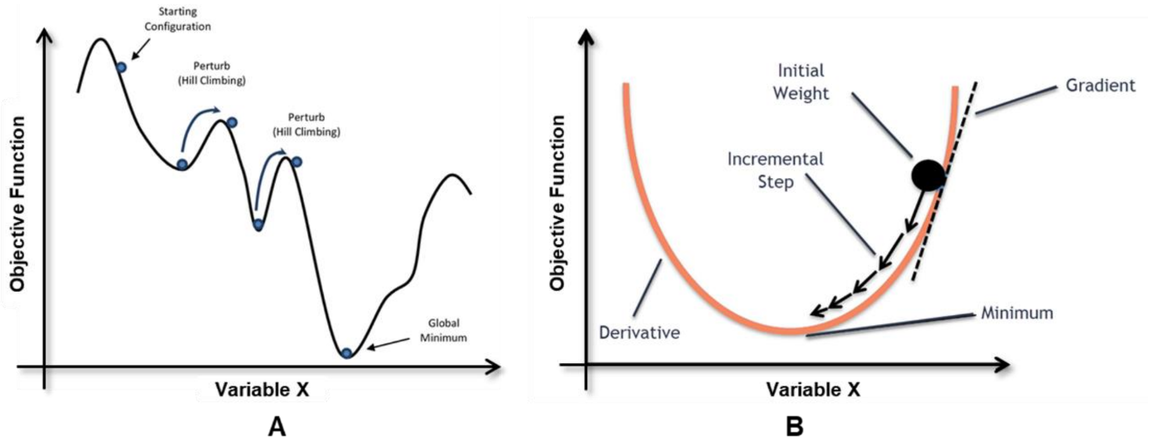


Fig. 2.4: Objective functions for optimization methods (A) and deep learning methods (B). Usually, objective functions in optimization methods contain many local minima/maxima, while non-convex objective functions (also called loss functions) are deliberately constructed in deep learning methods to make sure a local minimum is present to be found by gradient descent algorithms.

2.5. Competition or cooperation?

Optimization methods and DL methods are different categories for drug design. Optimization methods search for the global minimum (or maximum) of the objective functions, which are always a non-convex function and have many local optima (Fig. 2.4A). In contrast, DL models obtain the optimal parameters with a backpropagation algorithm by minimizing the loss function; these are usually constructed as convex functions to ensure a unique minimum to be sought by gradient descent algorithms (Fig. 2.4B). Traditionally, there were many successful cases in which the expected drug candidates were found through optimization methods. But these methods do not share a unified framework and users need to define some procedures manually case by case based on their experience. In recent years deep learning methods have come to the attention of researchers who have shown interest in applying them in drug design. Based on similar basic DL architectures, more and more promising methods have been proposed to learn knowledge from the training set efficiently and generate novel molecules automatically. By comparison, optimization methods are usually population-based, meaning each individual can be manipulated directly and conveniently to construct a pareto frontier for multiple objectives. Deep learning methods, however, are typically model-based, which can be used anywhere and the learned information can be passed on to other models through transfer learning.

However, current DL methods are still comparatively poor at handling the multiple objectives relevant for drug discovery; weighted summation is a common approach to tackle competitive objectives.

Table 2.3: Publicly and freely available data sources related to drug molecules

Name	Descriptions	URL
ChEMBL	Curated database of bioactive molecules with drug-like properties.	https://www.ebi.ac.uk/chembl/
PubChem	Collection of freely accessible chemical information, including chemical and physical properties, biological activities, safety and toxicity information, patents, *etc*.	https://pubchem.ncbi.nlm.nih.gov/
DrugBank	Bioinformatics and cheminformatics resource that combines detailed drug data with comprehensive drug target information	https://www.drugbank.ca/
SureChEMBL	database for chemical compounds in patents	https://www.surechembl.org
GDB	Combinatorically generated drug-like small molecule library	http://gdb.unibe.ch/
PDB	3D structure of Macromolecular Structures (including ligands binding to active site of targets)	https://www.rcsb.org/
QM9	Small organic molecules subset out of the GDB-17 with quantum chemical properties	http://www.quantum-machine.org/datasets/
ExCAPE-DB	An integrated chemogenomic dataset collected from publicly available databases including structure, target information and activity annotations	https://solr.ideaconsult.net/search/excape/
ZINC	Curated collection of commercially available chemical compounds	https://zinc15.docking.org/

The paradigm shift from the optimization methods to machine learning methods is mainly caused by the availability of large public databases and breakthroughs made in the field of deep learning in image and text generation. When optimization methods dominated the field of *de novo* drug design, there was little public data available as prior knowledge. Optimization methods focused on the objective functions, which were summarized based on a limited number of ligands, and the data wasonly used to provide the initial states or form the rules as constraints for molecule generation. In the age of big data public online databases (Table 2.3) such as ChEMBL [98,99], PubChem [100], ZINC [101], DrugBank [102,103], provide massive amounts of training data. Machine learning methods are now commonly used to extract useful information from this "big data" of drugs. Despite the current popularity of DL methods, it is worth noting that some researchers have questioned the performance of DL and benchmarked the performance between DL and other optimization methods. For example, Yoshikawa *et al.* employed a grammatical evolution to develop a SMILES-based drug design algorithm, called ChemGE, which generated molecules with high binding affinity. They compared their method with three other DL methods, including CVAE, GVAE and ChemTS. They found that with eight hours compute time, their method performed better than, or was comparable to DL methods. Similarly, Jensen proposed a graph-based GA approach for drug design which was shown to perform better than a SMILES-based RNN, the ChemTS, CVAE and GVAE with much lower computational cost.

Despite the differences in their mode of operation, some groups have tried to combine these two classes of methods for drug design. For example, an end-to-end model can map each molecule from discrete chemical space into a continuous latent space, *i.e.* the chemical structure can be converted into a numerical vector by the encoder. Such continuous representations are convenient for use in optimization and the resulting optima are subsequentially reconstructed into the expected molecules by the decoder. For example, Sattarov *et al.* applied a generative topographic mapping (GTM) technique, the probabilistic counterpart of self-organizing maps based on Bayesian learning, in the continuous space constructed by a VAE model [57]. GTM was convenient for visualization of the latent space in which target zones can be used for generating novel molecular structures by sampling. They succeeded in generating focused libraries of potential ligands toward the adenosine A_{2a} Receptor. In addition, Winter *et al.* constructed another end-to-end deep learning framework to construct a continuous space and exploited a PSO algorithm on this latent space. They were able to successfully generate ligands with a predicted high affinity to both EGFR and BACE1 [62].

2.6. Conclusion and perspective

In this review, we give a brief description of algorithms used in drug *de novo* design, divided in optimization methods on one hand and DL methods on the other hand. Traditionally, the drug design problem was always addressed as a combinatorial optimization problem. Hence optimization methods were dominant in drug design. With the rise of DL, more and more researchers shifted their interests from optimization algorithms to DL-based methods. The application of deep learning in drug *de novo* design caused a revolutionary pattern shift in drug discovery. However, DL methods have still a long way to go and traditional optimization algorithms still provide inspiration to improve the capability of drug *de novo* design. Currently, it is hard to say which kind of methods are dominant for all cases of drug design. Users should select methods based on their own conditions in practice. We also expect more sophisticated AI algorithms being proposed in the future to accelerate drug discovery

References

1. Polishchuk PG, Madzhidov TI, Varnek A (2013) Estimation of the size of drug-like chemical space based on GDB-17 data. J Comput Aided Mol Des 27 (8):675-679. doi:10.1007/s10822-013-9672-4
2. Macarron R, Banks MN, Bojanic D et al (2011) Impact of high-throughput screening in biomedical research. Nature reviews Drug discovery 10 (3):188-195. doi:10.1038/nrd3368
3. Giacomini KM, Krauss RM, Roden DM et al (2007) When good drugs go bad. Nature 446 (7139):975-977. doi:10.1038/446975a
4. Lounkine E, Keiser MJ, Whitebread S et al (2012) Large-scale prediction and testing of drug activity on side-effect targets. Nature 486 (7403):361-367. doi:10.1038/nature11159
5. Paul SM, Mytelka DS, Dunwiddie CT et al (2010) How to improve R&D productivity: the pharmaceutical industry's grand challenge. Nature reviews Drug discovery 9 (3):203-214. doi:10.1038/nrd3078
6. Kapetanovic IM (2008) Computer-aided drug discovery and development (CADDD): in silico-chemico-biological approach. Chem Biol Interact 171 (2):165-176. doi:10.1016/j.cbi.2006.12.006
7. Schneider G, Fechner U (2005) Computer-based de novo design of drug-like molecules. Nature reviews Drug discovery 4 (8):649-663. doi:10.1038/nrd1799
8. Chen H, Engkvist O, Wang Y et al (2018) The rise of deep learning in drug discovery. Drug discovery today. doi:10.1016/j.drudis.2018.01.039
9. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521 (7553):436-444. doi:10.1038/nature14539
10. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. Paper presented at the Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, Lake Tahoe, Nevada,
11. Goodfellow IJ, Pouget-Abadie J, Mirza M et al (2014) Generative Adversarial Networks. ArXiv:1406.2661
12. Silver D, Huang A, Maddison CJ et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587):484-489. doi:10.1038/nature16961
13. Gomez-Bombarelli R, Wei JN, Duvenaud D et al (2018) Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. ACS Cent Sci 4 (2):268-276. doi:10.1021/acscentsci.7b00572
14. Nicolaou CA, Brown N (2013) Multi-objective optimization methods in drug design. Drug Discov Today Technol 10 (3):e427-435. doi:10.1016/j.ddtec.2013.02.001
15. Sanchez-Lengeling B, Aspuru-Guzik A (2018) Inverse molecular design using machine learning: Generative models for matter engineering. Science 361 (6400):360-365. doi:10.1126/science.aat2663
16. van Westen GJP, Wegner JK, Ijzerman AP et al (2011) Proteochemometric modeling as a tool to design selective compounds and for extrapolating to novel targets. MedChemComm 2 (1):16-30. doi:10.1039/C0MD00165A
17. Rogers D, Hahn M (2010) Extended-connectivity fingerprints. Journal of chemical information and modeling 50 (5):742-754. doi:10.1021/ci100050t
18. von Lilienfeld OA (2013) First principles view on chemical compound space: Gaining rigorous atomistic control of molecular properties. International Journal of Quantum Chemistry 113 (12):1676-1689. doi:10.1002/qua.24375
19. Elton DC, Boukouvalas Z, Fuge MD et al (2019) Deep learning for molecular design—a review of the state of the art. Molecular Systems Design & Engineering. doi:10.1039/C9ME00039A
20. Noel OB, Andrew D (2018) DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of

Chemical Structures. doi:10.26434/chemrxiv.7097960.v1

21. Josep A-P, Simon Viet J, Oleksii P et al (2019) Randomized SMILES Strings Improve the Quality of Molecular Generative Models. doi:10.26434/chemrxiv.8639942.v2
22. Krenn M, Häse F, Nigam A et al (2019) SELFIES: a robust representation of semantically constrained graphs with an example application in chemistry. arXiv e-prints:arXiv:1905.13741
23. Emmerich MTM, Deutz AH (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Nat Comput 17 (3):585-609. doi:10.1007/s11047-018-9685-y
24. Mock WBT (2011) Pareto Optimality. In: Chatterjee DK (ed) Encyclopedia of Global Justice. Springer Netherlands, Dordrecht, pp 808-809. doi:10.1007/978-1-4020-9160-5_341
25. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: empirical results. Evol Comput 8 (2):173-195. doi:10.1162/106365600568202
26. Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. Trans Evol Comp 6 (2):182-197. doi:10.1109/4235.996017
27. Emmerich M, Beume N, Naujoks B An EMO Algorithm Using the Hypervolume Measure as Selection Criterion. In, Berlin, Heidelberg, 2005. Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, pp 62-76
28. Wang R, Gao Y, Lai L (2000) LigBuilder: A Multi-Purpose Program for Structure-Based Drug Design. Molecular modeling annual 6 (7):498-516. doi:10.1007/s0089400060498
29. Douguet D, Thoreau E, Grassy G (2000) A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. Journal of Computer-Aided Molecular Design 14 (5):449-466. doi:10.1023/A:1008108423895
30. Pegg SC, Haresco JJ, Kuntz ID (2001) A genetic algorithm for structure-based de novo design. J Comput Aided Mol Des 15 (10):911-933. doi:10.1023/a:1014389729000
31. Budin N, Majeux N, Tenette–Souaille C et al (2001) Structure-based ligand design by a build-up approach and genetic algorithm search in conformational space. Journal of Computational Chemistry 22 (16):1956-1970. doi:10.1002/jcc.1145
32. Vinkers HM, de Jonge MR, Daeyaert FF et al (2003) SYNOPSIS: SYNthesize and OPtimize System in Silico. Journal of medicinal chemistry 46 (13):2765-2773. doi:10.1021/jm030809x
33. Douguet D, Munier-Lehmann H, Labesse G et al (2005) LEA3D: a computer-aided ligand design for structure-based drug design. Journal of medicinal chemistry 48 (7):2457-2468. doi:10.1021/jm0492296
34. Dey F, Caflisch A (2008) Fragment-based de novo ligand design by multiobjective evolutionary optimization. Journal of chemical information and modeling 48 (3):679-690. doi:10.1021/ci700424b
35. van der Horst E, Marques-Gallego P, Mulder-Krieger T et al (2012) Multi-objective evolutionary design of adenosine receptor ligands. Journal of chemical information and modeling 52 (7):1713-1721. doi:10.1021/ci2005115
36. Lameijer EW, Kok JN, Back T et al (2006) The molecule evoluator. An interactive evolutionary algorithm for the design of drug-like molecules. Journal of chemical information and modeling 46 (2):545-552. doi:10.1021/ci050369d
37. Nicolaou CA, Apostolakis J, Pattichis CS (2009) De novo drug design using multiobjective evolutionary graphs. Journal of chemical information and modeling 49 (2):295-307. doi:10.1021/ci800308h
38. Fechner U, Schneider G (2006) Flux (1): a virtual synthesis scheme for fragment-based de novo design. Journal of chemical information and modeling 46 (2):699-707. doi:10.1021/ci0503560
39. Schneider G, Lee ML, Stahl M et al (2000) De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. J Comput Aided Mol Des 14 (5):487-494. doi:10.1023/a:1008184403558
40. Sengupta S, Bandyopadhyay S (2012) De novo design of potential RecA inhibitors using multi objective

optimization. IEEE/ACM Trans Comput Biol Bioinform 9 (4):1139-1154. doi:10.1109/TCBB.2012.35

41. Pearlman DA, Murcko MA (1996) CONCERTS: dynamic connection of fragments as an approach to de novo ligand design. Journal of medicinal chemistry 39 (8):1651-1663. doi:10.1021/jm950792l
42. Dean PM, Firth-Clark S, Harris W et al (2006) SkelGen: a general tool for structure-based de novo ligand design. Expert Opin Drug Discov 1 (2):179-189. doi:10.1517/17460441.1.2.179
43. Hartenfeller M, Proschak E, Schuller A et al (2008) Concept of combinatorial de novo design of drug-like molecules by particle swarm optimization. Chem Biol Drug Des 72 (1):16-26. doi:10.1111/j.1747-0285.2008.00672.x
44. Vikhar PA Evolutionary algorithms: A critical review and its future prospects. In: 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 22-24 Dec. 2016 2016. pp 261-265. doi:10.1109/ICGTSPICC.2016.7955308
45. Bäck T (1996) Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University Press, Inc.,
46. Mitchell M (1998) An Introduction to Genetic Algorithms. MIT Press,
47. Neill MO, Ryan C (2001) Grammatical evolution. IEEE Transactions on Evolutionary Computation 5 (4):349-358. doi:10.1109/4235.942529
48. Hansen N, Kern S Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In: Yao X, Burke EK, Lozano JA et al. (eds) Parallel Problem Solving from Nature - PPSN VIII, Berlin, Heidelberg, 2004// 2004. Springer Berlin Heidelberg, pp 282-291
49. Kennedy J, Eberhart R Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, 27 Nov.-1 Dec. 1995 1995. pp 1942-1948 vol.1944. doi:10.1109/ICNN.1995.488968
50. Oleksii P, Simon J, Panagiotis-Christos K et al (2019) A De Novo Molecular Generation Method Using Latent Vector Based Generative Adversarial Network. doi:10.26434/chemrxiv.8299544.v2
51. Putin E, Asadulaev A, Vanhaelen Q et al (2018) Adversarial Threshold Neural Computer for Molecular de Novo Design. Molecular pharmaceutics 15 (10):4386-4397. doi:10.1021/acs.molpharmaceut.7b01137
52. Blaschke T, Olivecrona M, Engkvist O et al (2018) Application of Generative Autoencoder in De Novo Molecular Design. Molecular informatics 37 (1-2). doi:10.1002/minf.201700123
53. Yang X, Zhang J, Yoshizoe K et al (2017) ChemTS: an efficient python library for de novo molecular generation. Sci Technol Adv Mater 18 (1):972-976. doi:10.1080/14686996.2017.1401424
54. Kang S, Cho K (2019) Conditional Molecular Design with Deep Generative Models. Journal of chemical information and modeling 59 (1):43-52. doi:10.1021/acs.jcim.8b00263
55. Griffiths R-R, Hernández-Lobato JM (2017) Constrained Bayesian Optimization for Automatic Chemical Design. eprint arXiv:170905501:arXiv:1709.05501
56. Merk D, Friedrich L, Grisoni F et al (2018) De Novo Design of Bioactive Small Molecules by Artificial Intelligence. Molecular informatics 37 (1-2). doi:10.1002/minf.201700153
57. Sattarov B, Baskin, II, Horvath D et al (2019) De Novo Molecular Design by Combining Deep Autoencoder Recurrent Neural Networks with Generative Topographic Mapping. Journal of chemical information and modeling 59 (3):1182-1196. doi:10.1021/acs.jcim.8b00751
58. Popova M, Isayev O, Tropsha A (2018) Deep reinforcement learning for de novo drug design. Sci Adv 4 (7):eaap7885. doi:10.1126/sciadv.aap7885
59. Polykovskiy D, Zhebrak A, Vetrov D et al (2018) Entangled Conditional Adversarial Autoencoder for de Novo Drug Discovery. Molecular pharmaceutics 15 (10):4398-4405. doi:10.1021/acs.molpharmaceut.8b00839
60. Segler MHS, Kogej T, Tyrchan C et al (2018) Generating Focused Molecule Libraries for Drug

Discovery with Recurrent Neural Networks. Acs Central Sci 4 (1):120-131. doi:10.1021/acscentsci.7b00512

61. Gupta A, Muller AT, Huisman BJH et al (2018) Generative Recurrent Networks for De Novo Drug Design. Molecular informatics 37 (1-2). doi:10.1002/minf.201700111
62. Winter R, Montanari F, Steffen A et al (2019) Efficient multi-objective molecular optimization in a continuous latent space. Chemical Science. doi:10.1039/C9SC01928F
63. Bjerrum EJ, Sattarov B (2018) Improving Chemical Autoencoder Latent Space and Molecular De Novo Generation Diversity with Heteroencoders. Biomolecules 8 (4). doi:10.3390/biom8040131
64. Lim J, Ryu S, Kim JW et al (2018) Molecular generative model based on conditional variational autoencoder for de novo molecular design. Journal of cheminformatics 10 (1):31. doi:10.1186/s13321-018-0286-7
65. Liu X, Ye K, van Vlijmen HWT et al (2019) An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine A2A receptor. Journal of cheminformatics 11 (1):35. doi:10.1186/s13321-019-0355-6
66. Olivecrona M, Blaschke T, Engkvist O et al (2017) Molecular de-novo design through deep reinforcement learning. Journal of cheminformatics 9 (1):48. doi:10.1186/s13321-017-0235-x
67. Zhou Z, Kearnes S, Li L et al (2018) Optimization of Molecules via Deep Reinforcement Learning. eprint arXiv:181008678:arXiv:1810.08678
68. Lima Guimaraes G, Sanchez-Lengeling B, Outeiral C et al (2017) Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. arXiv e-prints:arXiv:1705.10843
69. Putin E, Asadulaev A, Ivanenkov Y et al (2018) Reinforced Adversarial Neural Computer for de Novo Molecular Design. Journal of chemical information and modeling 58 (6):1194-1204. doi:10.1021/acs.jcim.7b00690
70. Dai H, Tian Y, Dai B et al (2018) Syntax-Directed Variational Autoencoder for Structured Data. arXiv e-prints
71. Kusner MJ, Paige B, Hernández-Lobato JM (2017) Grammar Variational Autoencoder. eprint arXiv:170301925:arXiv:1703.01925
72. Skalic M, Jimenez J, Sabbadin D et al (2019) Shape-Based Generative Modeling for de Novo Drug Design. Journal of chemical information and modeling 59 (3):1205-1214. doi:10.1021/acs.jcim.8b00706
73. Aumentado-Armstrong T (2018) Latent Molecular Optimization for Targeted Therapeutic Design. eprint arXiv:180902032:arXiv:1809.02032
74. Simonovsky M, Komodakis N (2018) GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders. eprint arXiv:180203480:arXiv:1802.03480
75. Liu Q, Allamanis M, Brockschmidt M et al (2018) Constrained Graph Variational Autoencoders for Molecule Design. eprint arXiv:180509076:arXiv:1805.09076
76. You J, Liu B, Ying R et al (2018) Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. eprint arXiv:180602473:arXiv:1806.02473
77. Jin W, Barzilay R, Jaakkola T (2018) Junction Tree Variational Autoencoder for Molecular Graph Generation. eprint arXiv:180204364:arXiv:1802.04364
78. Popova M, Shvets M, Oliva J et al (2019) MolecularRNN: Generating realistic molecular graphs with optimized properties. eprint arXiv:190513372:arXiv:1905.13372
79. Bradshaw J, Paige B, Kusner MJ et al (2019) A Model to Search for Synthesizable Molecules. eprint arXiv:190605221:arXiv:1906.05221
80. Stahl N, Falkman G, Karlsson A et al (2019) Deep Reinforcement Learning for Multiparameter Optimization in de novo Drug Design. Journal of chemical information and modeling.

doi:10.1021/acs.jcim.9b00325

81. Miljanovic M (2012) Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction. Indian Journal of Computer Science and Engineering 3
82. Graves A, Liwicki M, Fernández S et al (2009) A Novel Connectionist System for Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5):855-868. doi:10.1109/TPAMI.2008.137
83. Sak H, Senior A, Beaufays F (2014) Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH:338-342
84. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9 (8):1735-1780
85. Chung J, Gulcehre C, Cho K et al (2014) Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv:1412.3555
86. Segler MHS, Kogej T, Tyrchan C et al (2018) Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. ACS Cent Sci 4 (1):120-131. doi:10.1021/acscentsci.7b00512
87. Sumita M, Yang X, Ishihara S et al (2018) Hunting for Organic Molecules with Artificial Intelligence: Molecules Optimized for Desired Excitation Energies. ACS Cent Sci 4 (9):1126-1133. doi:10.1021/acscentsci.8b00213
88. Arus-Pous J, Blaschke T, Ulander S et al (2019) Exploring the GDB-13 chemical space using deep generative models. Journal of cheminformatics 11 (1):20. doi:10.1186/s13321-019-0341-z
89. Kramer MA (1991) Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal 37 (2):233-243. doi:10.1002/aic.690370209
90. Kingma D, Welling M (2014) Auto-Encoding Variational Bayes.
91. Doersch C (2016) Tutorial on Variational Autoencoders. arXiv e-prints:arXiv:1606.05908
92. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. J Artif Int Res 4 (1):237-285
93. Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8 (3):229-256. doi:10.1007/BF00992696
94. Arjovsky M, Chintala S, Bottou L (2017) Wasserstein GAN. arXiv e-prints:arXiv:1701.07875
95. The GAN Zoo. https://github.com/hindupuravinash/the-gan-zoo.
96. De Cao N, Kipf T (2018) MolGAN: An implicit generative model for small molecular graphs. eprint arXiv:180511973:arXiv:1805.11973
97. Makhzani A, Shlens J, Jaitly N et al (2015) Adversarial Autoencoders. arXiv e-prints:arXiv:1511.05644
98. Gaulton A, Bellis LJ, Bento AP et al (2012) ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic acids research 40 (Database issue):D1100-1107. doi:10.1093/nar/gkr777
99. Mendez D, Gaulton A, Bento AP et al (2019) ChEMBL: towards direct deposition of bioassay data. Nucleic acids research 47 (D1):D930-D940. doi:10.1093/nar/gky1075
100. Wang Y, Xiao J, Suzek TO et al (2009) PubChem: a public information system for analyzing bioactivities of small molecules. Nucleic acids research 37 (Web Server issue):W623-633. doi:10.1093/nar/gkp456
101. Sterling T, Irwin JJ (2015) ZINC 15--Ligand Discovery for Everyone. Journal of chemical information and modeling 55 (11):2324-2337. doi:10.1021/acs.jcim.5b00559
102. Wishart DS, Feunang YD, Guo AC et al (2018) DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic acids research 46 (D1):D1074-D1082. doi:10.1093/nar/gkx1037
103. Wishart DS, Knox C, Guo AC et al (2008) DrugBank: a knowledgebase for drugs, drug actions and drug targets. Nucleic acids research 36 (Database issue):D901-906. doi:10.1093/nar/gkm958

Chapter 3

An exploration strategy improves the diversity of *de novo* ligands using deep reinforcement learning: a case for the adenosine A_{2A} receptor

Xuhan Liu, Kai Ye, Herman W. T. van Vlijmen, Adriaan P. IJzerman and Gerard J. P. van Westen*. Journal of Cheminformatics (2019). Https://doi.org/10.1186/s13321-019-0355-6

Abstract

Over the last five years deep learning has progressed tremendously in both image recognition and natural language processing. Now it is increasingly applied to other data rich fields. In drug discovery, recurrent neural networks (RNNs) have been shown to be an effective method to generate novel chemical structures in the form of SMILES. However, ligands generated by current methods have so far provided relatively low diversity and do not fully cover the whole chemical space occupied by known ligands. Here, we propose a new method (DrugEx) to discover *de novo* drug-like molecules. DrugEx is an RNN model (generator) trained through a special exploration strategy integrated into reinforcement learning. As a case study we applied our method to design ligands against the adenosine A_{2A} receptor. From ChEMBL data, a machine learning model (predictor) was created to predict whether generated molecules are active or not. Based on this predictor as the reward function, the generator was trained by reinforcement learning without any further data. We then compared the performance of our method with two previously published methods, REINVENT and ORGANIC. We found that candidate molecules our model designed, and predicted to be active, had a larger chemical diversity and better covered the chemical space of known ligands compared to the state-of-the-art.

Keywords: deep learning; adenosine receptors; cheminformatics; reinforcement learning; exploration strategy.

3.1. Introduction

G Protein-Coupled Receptors (GPCRs) are the largest family of cell membrane-bound proteins [1], containing about 800 members encoded by approximately 4% of human genes. GPCRs are central to a large number of essential biological processes, including cell proliferation, cell survival, and cell motility [2]. Currently, GPCRs form the main target of approximately 34% of all FDA approved drugs [3]. One of the most extensively studied GPCRs is the human adenosine A_{2A} receptor (A_{2A}AR), which has been shown to be a promising drug target for Parkinson's disease, cardiovascular diseases, and inflammatory disorders [4]. Multiple crystal structures with different ligands have been resolved [5,6], and data on the biological activity of thousands of chemical compounds against the receptor was made available in the public ChEMBL database [7]. Considering the amount of data available and our in-house expertise we exploited machine learning methods to design novel ligands with predicted affinity for the A_{2A}AR.

Over the last years, deep learning (DL) has been at the forefront of great breakthroughs in the field of artificial intelligence and its performance even surpassed human abilities for image recognition and natural language processing [8]. Since then, deep learning is gradually being applied to other data rich fields [9,10]. In drug discovery DL has been used to construct quantitative structure-activity relationship (QSAR) models [11] to predict the properties of chemical compounds, such as toxicity, partition coefficient, affinity for specific targets, *etc.* [12,13]. Most commonly pre-defined descriptors such as Extended Connectivity Fingerprint (ECFP) [14] were used as input to construct fully-connected neural networks [15]. More recently studies were published using other methods wherein neural networks extract the descriptor from chemical structures automatically and directly, such as Mol2Vec [16], DruGAN [17], GraphConv [18], *etc*.

In addition to these *prediction* applications, DL can also be used in chemical structure *generation* [13]. Gupta *et al*. constructed a recurrent neural network (RNN) model to learn the syntax of the SMILES notation and generate novel SMILES for molecules representation [19]. In addition, Olivecrona *et al.* combined RNNs and reinforcement learning (RL) to generate SMILES formatted molecules that have required chemical and biological properties [20]. RL has been instrumental in the construction of “AlphaGo” designed by DeepMind, which defeated one of the best human Go players [21]. Finally, similar to generative adversarial networks (GANs) for generating images [22], Benjamin *et al.* exploited the GAN for a sequence generation model [23] to generate molecules with multi-objective optimization [24].

In order to maximize the chance to find interesting hits for a given target, generated drug candidates should a) be chemically diverse, b) possess biological activity, and c) contain similar physicochemical properties or chemical scaffolds to already known ligands [25]. Although several groups have studied the application of DL for generating molecules as drug candidates, most current generative models cannot satisfy all of these three conditions simultaneously [26]. Considering the variety of the structure and function of GPCRs and the huge space of drug candidates, it is impossible to enumerate all possible virtual molecules in advance [27]. Here we aimed to discover *de novo* drug-like molecules against A_{2A}AR by our proposed new method DrugEx in which an exploration strategy was integrated into a RL model. The integration of this function ensured that our model generated candidate molecules similar to known ligands of A_{2A}AR with great diversity and predicted affinity for the A_{2A}AR. All python code for this study is freely available at http://github.com/XuhanLiu/DrugEx.

3.2. Dataset and methods

3.2.1. Data source

Drug-like molecules were collected from the ZINC database (version 15) [28]. We randomly chose approximately one million SMILES formatted molecules that met the following criteria: $-2 < \log P < 6$ and $200 <$ molecular weight (MW) < 600. The dataset (named *ZINC* hereafter) finally contained 1,018,517 molecules, and was used for SMILES syntax learning. Furthermore, we extracted the known ligands for the A_{2A}AR (ChEMBL identifier: CHEMBL251) from ChEMBL (version 23) [29]. If multiple measurements for the same ligands existed, the average pCHEMBL value (pKi or pIC50 value) was calculated and duplicate items were removed. If the pCHEMBL value < 6.5 or the compound was annotated as “Not Active” it was regarded as a negative sample; otherwise, it was regarded as a positive sample. In the end this dataset (named as *A2AR*) contained 2,420 positive samples and 2,562 negative samples.

3.2.2. Prediction model (QSAR)

Binary classification through QSAR modelling was used as prediction task. Input data for the model were ECFP6 fingerprints with 4096 bits calculated by the RDKit Morgan Fingerprint algorithm with a three-bond radius [30]. Hence, each molecule in the dataset was transformed into a 4096D vector. Model output value was the probability whether a given chemical compound was active based on this vector. Four algorithms were benchmarked for model construction, Random Forest (RF), Support Vector Machine (SVM), Naïve Bayesian (NB), and Deep Neural Network (DNN). The RF, SVM and NB models were implemented through Scikit-Learn [31], and DNN through PyTorch [32]. In RF, the number of trees was set as 1000 and split criterion was “*gini*”. In SVM, a radial basis function (RBF) kernel was used and the parameter space of C and γ were set as $[2^{-5}, 2^{15}]$ and $[2^{-15}, 2^{5}]$, respectively. In DNN, the architecture contained three hidden layers activated by rectified linear unit (ReLU) between input and output layers (activated by sigmoid function), the number of neurons were 4096, 8000, 4000, 2000 and 1 for each layer. With 100 epochs of training process 20% of hidden neurons were randomly dropped out between each layer. The binary cross entropy was used to construct the loss function and optimized by Adam [33] with a learning rate of 10^{-3}. The AUC of ROC curves was calculated to compare their mutual performance.

3.2.3. Generative model

Starting from the SMILES format, each molecule in the *ZINC* dataset was split into a series of tokens, standing for different types of atoms and bonds. Then, all tokens existing in this dataset were collected to construct the SMILES vocabulary. The final vocabulary contained 56 tokens (Table S3.1) which were selected and arranged sequentially into valid SMILES sequence following the correct grammar.

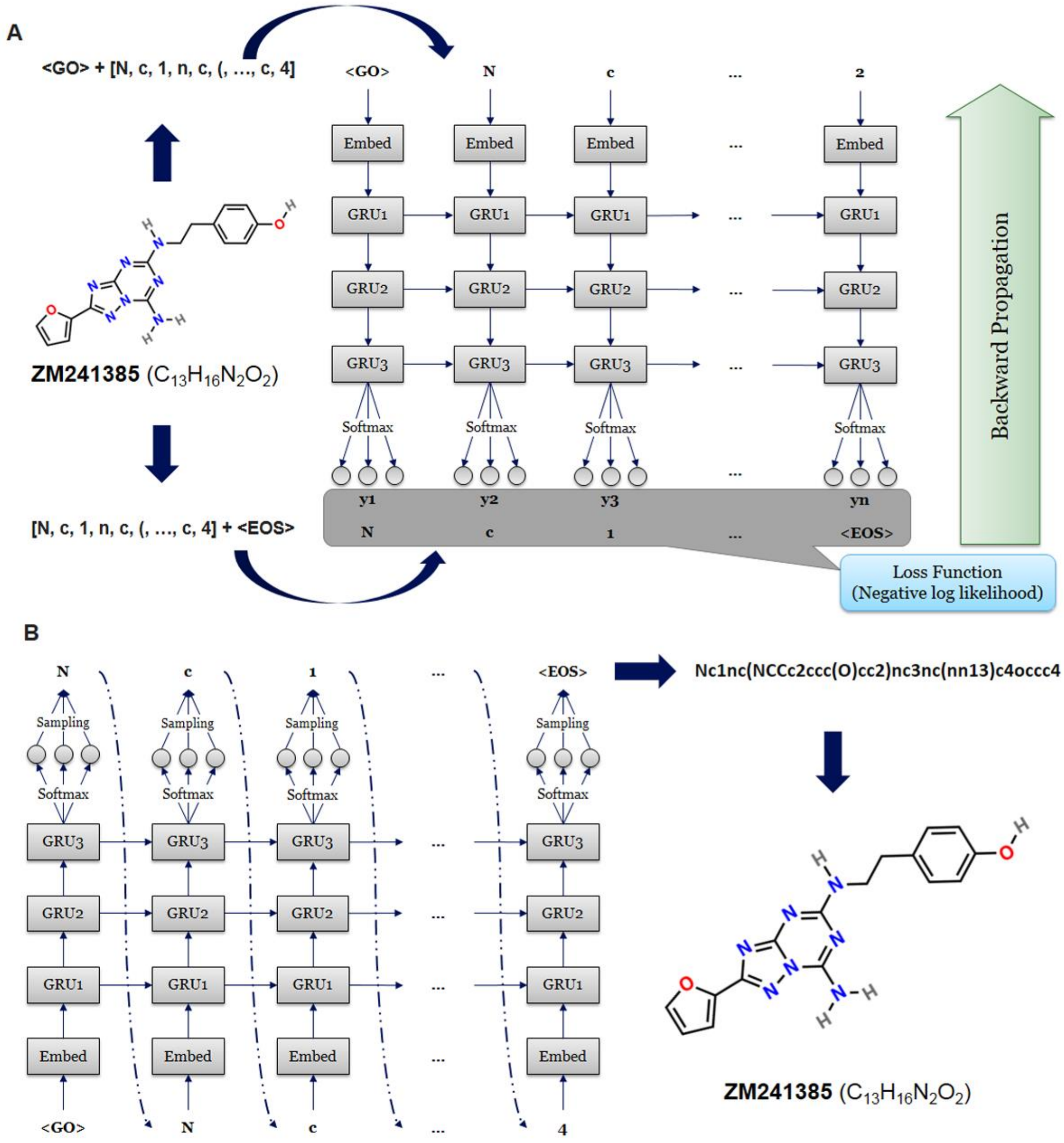

A
<GO> + [N, c, 1, n, c, (, …, c, 4]
ZM241385 ($C_{13}H_{16}N_2O_2$)
[N, c, 1, n, c, (, …, c, 4] + <EOS>
<GO>
N
c
2
Embed
GRU1
GRU2
GRU3
Softmax
y1
y2
y3
yn
N
c
1
<EOS>
Backward Propagation
Loss Function
(Negative log likelihood)
B
N
c
1
<EOS>
Sampling
Softmax
GRU3
GRU2
GRU1
Embed
<GO>
N
c
4
Nc1nc(NCCc2ccc(O)cc2)nc3nc(nn13)c4occc4
ZM241385 ($C_{13}H_{16}N_2O_2$)

Fig. 3.1: Architecture of recurrent neural networks for training and sampling processes with A_{2A}AR antagonist ZM241385 as an example. (A) In the training process of RNNs, each molecule is decomposed to a series of tokens and then taken as input. Subsequently, the input and output are combined with a start token and an end token, respectively. (B) Beginning with the start token "GO", the model calculates the probability distribution of each token in the vocabulary. For each step, one of the available tokens is randomly chosen based on the probability distribution and is again received by RNNs as input to calculate the new probability distribution for the next step. This process will end if the end token "EOS" is sampled or the number of steps equals 100.

The RNN model constructed for sequence generation contained six layers: one input layer, one embedding layer, three recurrent layers and one output layer (Fig. 3.1). After being represented by a sequence of tokens, molecules can be received as categorical features by the input layer. In the embedding layer, vocabulary size, and embedding dimension were set to 56 and 128, meaning each token could be transformed into a 128d vector. For the recurrent layer, gated recurrent unit (GRU) [34] was used as the recurrent cell with 512 hidden neurons. The output at each position was the probability that determined which token in the vocabulary would be chosen to construct the SMILES string.

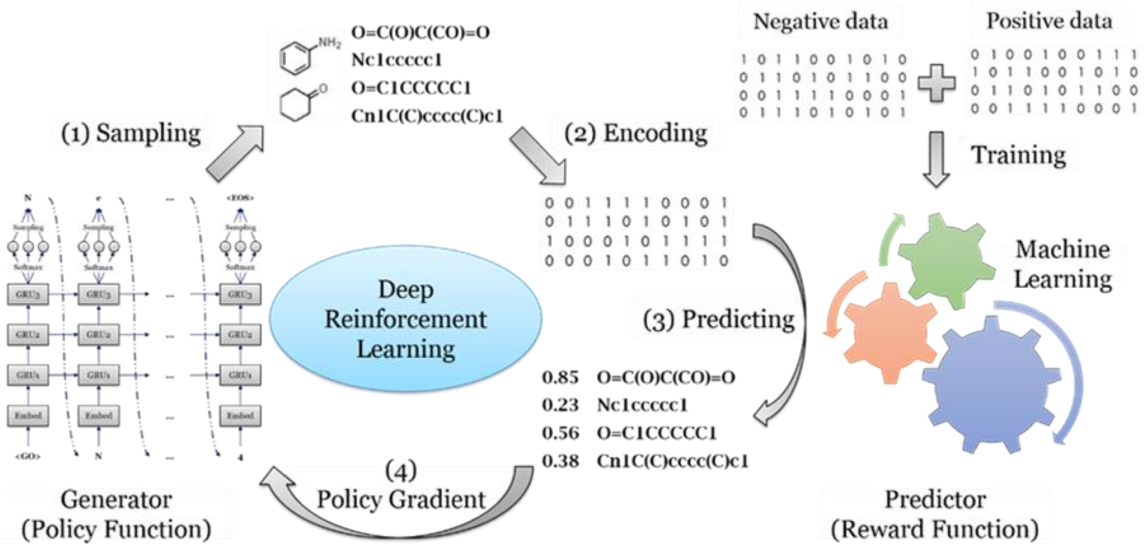


Fig. 3.2: The workflow of deep reinforcement learning. For each loop, it contains several steps: (1) a batch of SMILES sequences was sampled by the generator, which had been initialized by a pre-trained model; (2) each generated molecule represented by this SMILES format was encoded into a fingerprint; (3) a probability score of activity on the A_{2A}AR was assigned to each molecule, calculated by the QSAR model which had been trained in advance; (4) all of the generated molecules and their scores were sent back for training of the generator with the policy gradient method.

During the training process we put the start token at the beginning of a batch of data as input and the end token at the end of the same batch of data as output. This ensures that the generative network could choose correct tokens based on the sequence it had generated (Fig. 3.1A). A negative log likelihood function was used to construct the loss function to guarantee that the token in the output sequence had the largest probability to be chosen after being trained. In order to optimize the parameters of the model, the Adam algorithm [33] was used for optimization of loss function. Here, the learning rate was set at 10^{-3}, batch size was 500, and training steps set at 1000 epochs.

3.2.4. Reinforcement learning

SMILES sequence construction under the RL framework can be viewed as a series of decision-making processes (Fig. 3.2). At each step, the model determines the optimal token from the vocabulary based on the generated sequence in previous steps. However, the pure RNN model cannot guarantee that the percentage of desired molecules (i.e. biologically active on the A_{2A}AR) being generated is as large as possible. To solve this problem RL is an appropriate method because it increases the probability of those molecules with higher rewards and avoids generating those molecules with lower rewards. We regarded the generator as the policy function and the predictor as the reward function. The generator G_θ was updated by employing a policy gradient on the basis of the expected end reward received from the predictor Q. The objective function could be designated as generating a sequence from the start state to maximize the expected end reward [23].

$$J(\theta) = E[R(y_{1:T})|\theta] = \sum_{t=1}^{T} G_\theta(y_t|y_{1:t-1}) \cdot (Q(y_{1:T}) - \beta)$$

Here R is the reward for a complete sequence which is given by the prediction model Q; the generative model G_θ can be regarded as policy function to determine the probability of each token from the vocabulary to be chosen. The parameter β was the baseline of the reward, meaning that if the reward score was not larger than the baseline, the model would take it as a minus score or punishment. The goal of the generative model is to construct a sequence which can obtain the highest score judged by the predictor.

3.2.5. Exploration strategy

In order to improve the diversity of generated molecules, the token selection was not only determined by the generator constructed by the RNN model as described above, but also by a second fixed pre-trained RNN model (Fig. 3.3). The RNN requiring training is deemed the 'exploitation network' (G_θ) and the fixed RNN (not requiring training) is deemed the 'exploration network' (G_φ). Both had an identical network architecture. We define “exploring rate” (ε) in [0.0, 1.0] to determine which fraction of steps was determined by the exploration network. During the training process, each SMILES sequence was generated through the collaboration of these two RNNs. At each step a random number in [0.0, 1.0] was generated. If the value was smaller than ε, the G_φ would determine which token to be chosen, and vice versa. After the training process was finished, we removed G_φ and only G_θ was left as the final model of DrugEx for molecule generation.

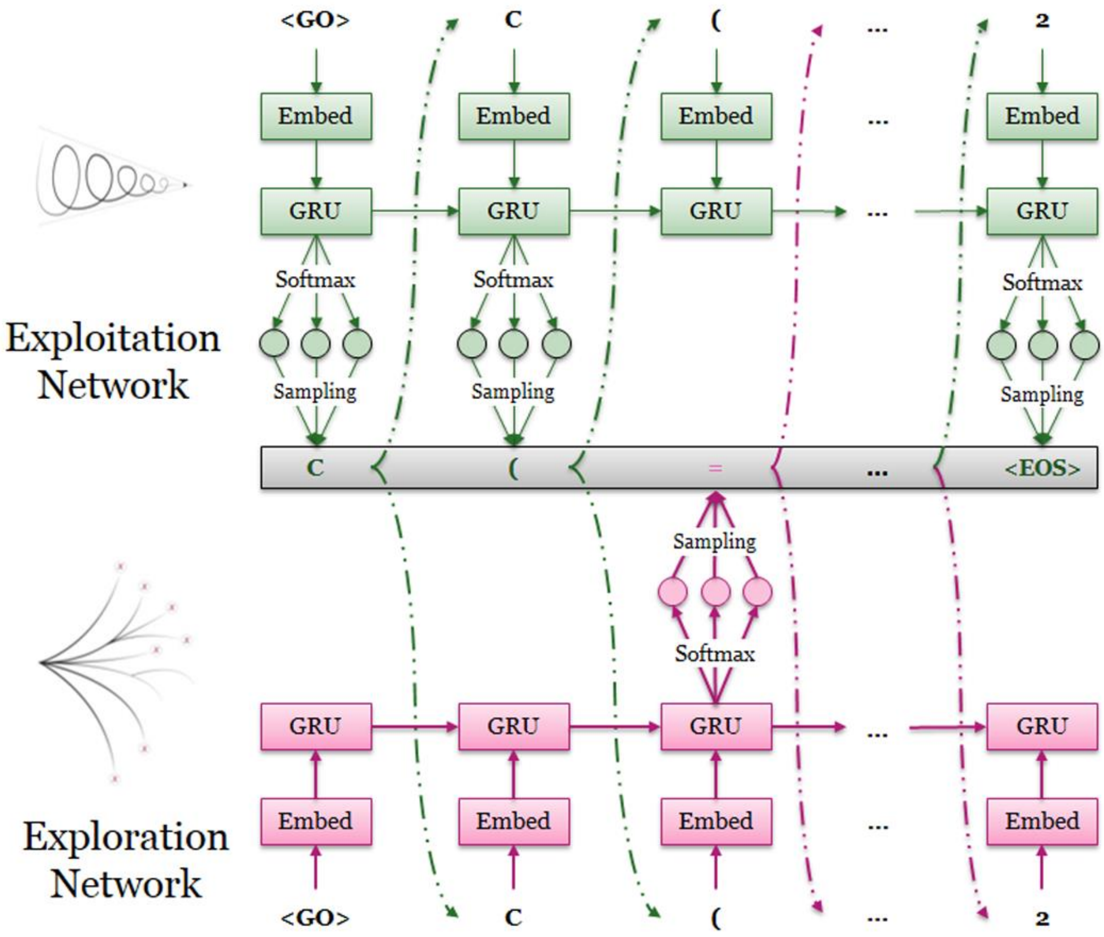


Fig. 3.3: Molecule generation with the assistance of the exploration strategy during the training process. For each step of token selection, a random variable was generated between 0 and 1. If the value is larger than a pre-set threshold (exploring rate, ε), the probability distribution is determined by the current generator (exploitation network, G_θ). Otherwise, it was determined by the exploration network (G_φ).

3.2.6. Molecular diversity

The Tanimoto-similarity was used for measuring the similarity of molecules. Given two compounds a and b and their ECFP6 fingerprints m_a and m_b, the Tanimoto-similarity is defined as:

$$T_s(a,b) = \frac{|m_a \cap m_b|}{|m_a \cup m_b|}$$

where $|m_a \cap m_b|$ represents the number of common fingerprint bits, and $|m_a \cup m_b|$ donates the total number of fingerprint bits. The Tanimoto-distance is defined as:

$$T_d(a,b) = 1 - T_s(a,b)$$

Similar to Benhenda [26], the diversity I of a set of molecules A (with size of |A|) is defined as the average of the Tanimoto-distance of molecules of every pair of molecules:

$$I(A) = \frac{1}{|A|^2} \sum_{(a,b) \in A \times A} T_d(a,b)$$

In a given set of molecules, the less similar each two molecules are, the larger the value of its diversity will be.

3.3. Results and discussion

3.3.1. Performance of predictors

All molecules in the *A2AR* dataset were used for training the QSAR models, after being transformed into ECFP6 fingerprints. We then tested the performance of these different algorithms with five-fold cross validation of which the ROC curves are shown in Fig. 3.4. The RF model achieved the highest value of AUC, MCC, Sensitivity, and Accuracy, despite its Specificity being slightly lower than DNN. Hence this model was chosen as our predictor whose output would be regarded as the reward for the generator in RL. In our previous study [15], the performance of DNN was better than RF on the chemical space of the whole ChEMBL database. A possible reason for this difference can be that the size of the *A2AR* dataset and chemical diversity was much smaller than ChEMBL as a whole. This has a negative influence on DNN which had more parameters to be optimized than RF. Selecting the predictor was a critical step in this study, as this model would be used to determine whether the following generated molecules were active or inactive.

3.3.2. SMILES syntax learning

For the training of RNNs all molecules in the *ZINC* dataset were used as training set after being decomposed into the tokens which belonged to our vocabulary set. Here, we defined that a SMILES sequence was valid if it could be parsed by RDKit. During the training process, the percentage of valid SMILES sequence through 1,000 times sampling was calculated and was then recorded with the value of the loss function at each epoch (Fig.

3.5A). After about 300 epochs, the loss function had converged, indicating the model was trained well.

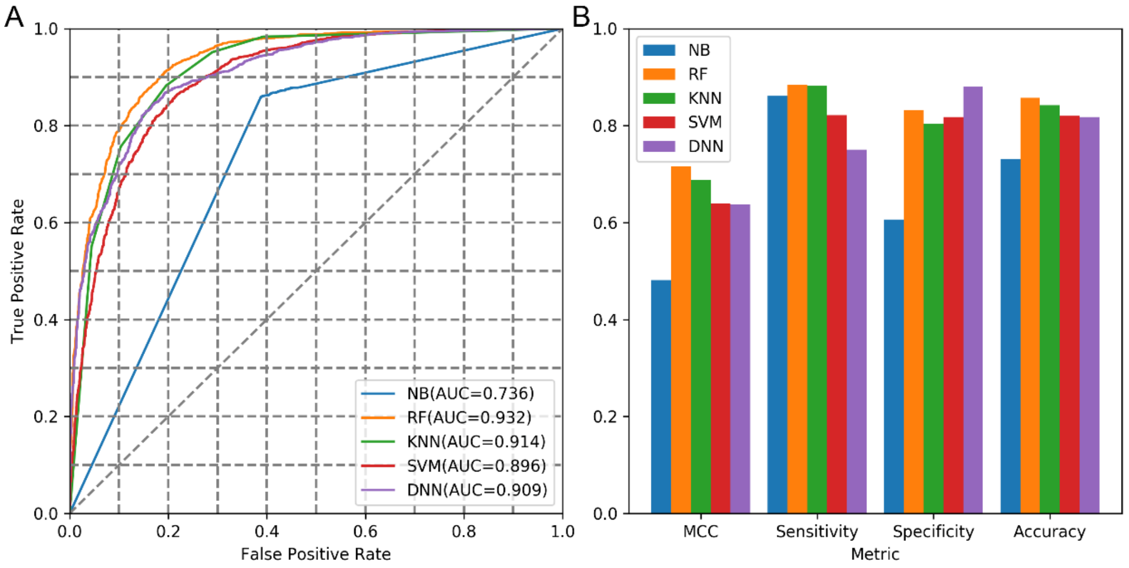


Fig. 3.4: Performance of five different machine learning models based on five-fold cross validation in A2AR dataset with different metrics, including AUC of ROC curve (A), MCC, Sensitivity, Specificity and Accuracy values (B). Except for specificity RF achieved highest scores among these models based on such measurements.

Subsequently, we sampled 10,000 SMILES sequences based on this well-trained model and found that 93.88% of these sequences were grammatically correct SMILES. We then compared some properties of these generated molecules with those in the training set, including number of hydrogen bond donors/acceptors, rotatable bonds, and different kind of ring systems (Fig. 3.6A). The distribution of these properties in the generated molecules highly resembles the molecules in the *ZINC* dataset. The logP ~ MW plot (Fig. 3.7A) shows that most generated molecules were drug-like molecules and cover the vast majority of the square space occupied by the *ZINC* dataset. Besides these eight properties, we also calculated 11 other physicochemical properties (including topological polar surface area, molar refractivity, the fraction of sp^3 hybridized carbon atoms and number of amide bonds, bridgehead atoms, heteroatoms, heavy atoms, spiroatoms, rings, saturated rings, valence electrons) to form 19D physicochemical descriptors (PhysChem). In addition, principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) were employed for dimension reduction and chemical space visualization with the PhysChem and ECFP6 of these molecules, respectively. We found that generated molecules covered almost the whole region occupied by molecules in the *ZINC* dataset (Fig. 3.7 B and C) although the number of these generated molecules was less than 1% of the number of molecules in the *ZINC* dataset.

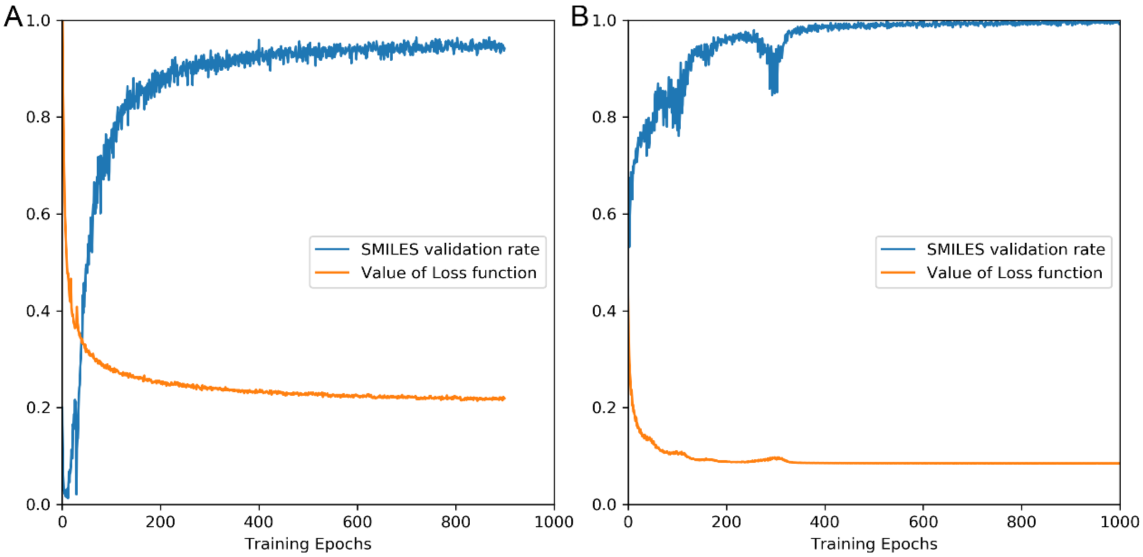


Fig. 3.5: The value of loss function and percentage of valid SMILES sequence during the pre-trained process on *ZINC* dataset (B) and fine-tuned process on *A2AR* dataset (B). The model was well pre-trained after 300 epochs and these two values converged to 0.19 and 93.88%, respectively. The performance of the fine-tuned model converged after 400 epochs with the two values reaching 0.09 and 0.99, respectively.

Subsequently we used the *A2AR* dataset to fine-tune this pre-trained model with 1,000 epochs (Fig. 3.5B). After sampling another 10,000 times, we performed the same comparison with the *A2AR* dataset with respect to the properties mentioned above (Fig. 3.6B) and investigated the chemical space represented by logP ~ MW (Fig. 3.7D), first two components of PCA (Fig. 3.7E) and t-SNE (Fig. 3.7F), yielding results similar to the model without fine-tuning but then focused on the *A2AR* chemical space. These results prove that RNN is an appropriate method to learn the SMILES grammar and to construct molecules similar to the ligands in the training set, which has also been shown in other work [35,19].

3.3.3. Conditional SMILES generation

The RNN model trained on the *ZINC* dataset was used as an initial state for the policy gradient in RL. After the training process of RL converged, 10,000 SMILES sequences were generated for performance evaluation. However, after removal of duplicates in these sequences, only less than 10 unique molecules were left which were similar to compounds in the *A2AR* dataset. When checking the log file of the training process, we noticed that these duplicated sequences were frequently sampled at each epoch and its duplication rate increased gradually. In order to decrease the bias caused by these molecules with high frequency, we removed all duplicated sequences sampled at each epoch for training with the policy gradient. We found that almost all of the molecules generated according to this procedure were located outside of the drug-like region with regard to logP ~ MW plot (Figure S1). This problem might be caused by the bias of the predictor. ECFP is a substructure-based fingerprint, implying that as long as the molecule contains some critical substructures, it will be prone to be predicted as active. That was the reason why generated SMILES sequences contained a large number of repetitive motifs. Several research groups have made improvements to guarantee that the final model has ability to generate drug-like candidate molecules [24,20]. In the next section, we will describe our proposed method, "DrugEx" by integrating an exploration strategy to solve this problem and compare it to existing methods.

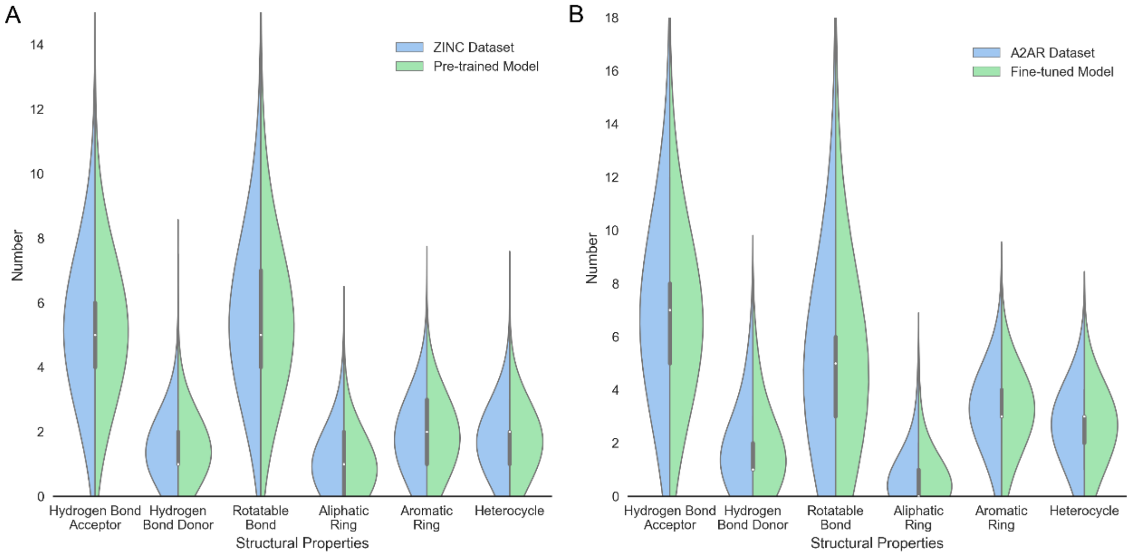


Fig. 3.6: Comparison of the properties of generated molecules by the pre-trained (A) and fine-tuned models (B) and molecules in *ZINC* dataset (A) and *A2AR* dataset (B), respectively. These properties included the number of acceptor/donor of hydrogen bonds, rotatable bonds, aliphatic rings, aromatic rings, and heterocycles.

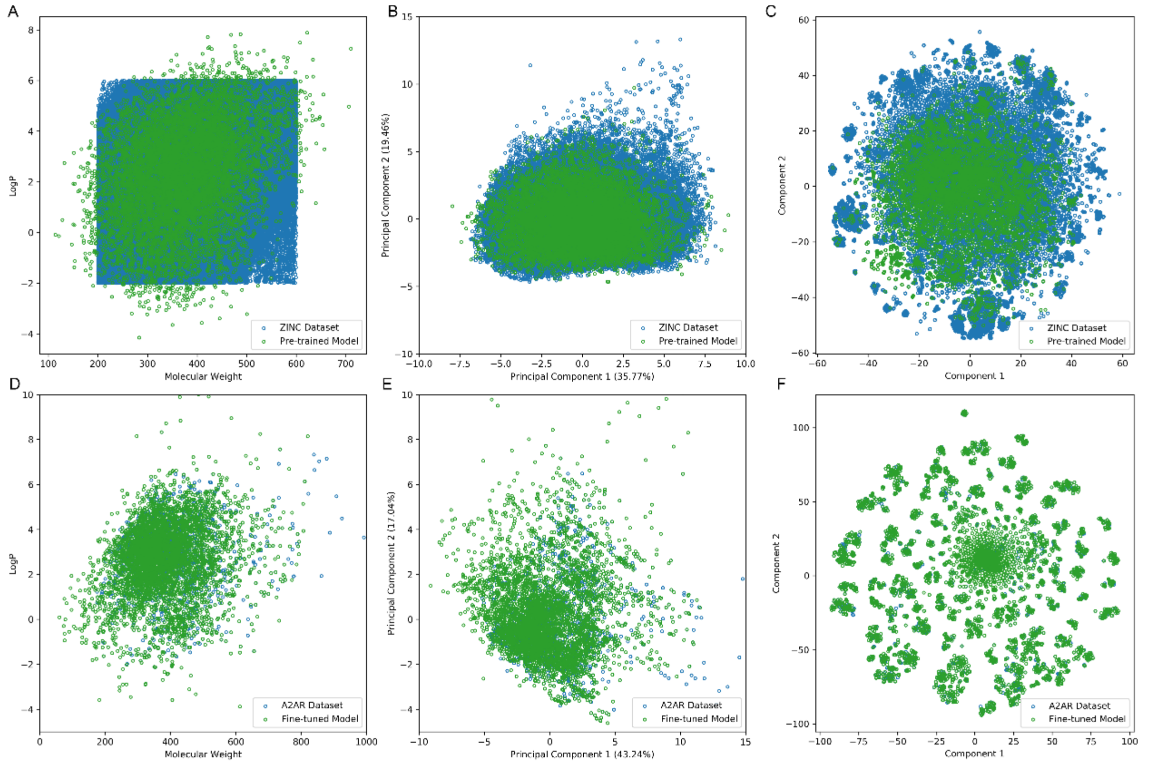


Fig. 3.7: The chemical space of generated molecules by pre-trained models with *ZINC* dataset (A-C) and fine-tuned model with *A2AR* dataset (D-F). The chemical space was represented by either logP ~ MW (A, D) and first two components in PCA on PhysChem descriptors (C, E) and t-SNE on ECFP6 fingerprints (D, F).

3.3.4. Exploration strategy

During the training process, the generated sequence is determined by both the G_θ and the G_φ where ε determines how many contributions the G_φ made. The G_φ and G_θ were both initialized by the pre-trained RNN model on the *ZINC* dataset. The G_φ was fixed and only parameters in the G_θ were updated. In order to optimize parameters, the parameter space was designated [0.01, 0.1] and [0.0, 0.1] for ε and β, respectively. After the model converged at 200 epochs (Fig. 3.8A), the performance of these models was evaluated subsequently based on 10,000 sampled sequences. Firstly, it was found that the number of duplicate SMILES notations was reduced dramatically and almost all SMILES notations represented drug-like molecules (Fig. 3.9A, 10D). Table 3.1 shows that when ε was increased, the model generated fewer active ligands for A_{2A}AR but the diversity of generated molecules (represented as unique desired SMILES) increased significantly. It was also observed that with higher ε, the distribution of different kinds of ring systems in the generated desired molecules became more similar to the known active ligands in the

A2AR dataset (Fig. 3.9A). In order to further investigate the effect of ε, we also tested a range of different values as [0.01, 0.05, 0.10, 0.15, 0.20, 0.25] and the results are shown in Figure S2. The G_φ can hence help the model produce more molecules similar to known active ligands of the given target but not identical to them. At higher ε, the baseline can help the model improve the average score and generate more desired molecules. However, this effect was not significant at lower values of ε. It is worth noticing in this study that if the baseline was larger than 0.1, the training process of the generative model did not converge.

Table 3.1: Comparison of the performance of the different methods

	DrugEx (Pre-trained)				**DrugEx (Fine-tuned)**				**REINVENT**	**ORGANIC**	**Pre-trained**	**Fine-tuned**
ε	0.01	0.01	0.1	0.1	0.01	0.01	0.1	0.1	--	--	--	--
β	0.0	0.1	0.0	0.1	0.0	0.1	0.0	0.1	--	--	--	--
Valid SMILES	98.3%	98.9%	95.9%	98.8%	99.1%	99.0%	98.2%	97.5%	98.8%	99.8%	93.9%	96.2%
Desired SMILES	97.5%	98.0%	74.6%	80.9%	98.3%	98.5%	94.4%	94.5%	98.2%	99.8%	0.7%	47.9%
Unique SMILES	96.5%	96.3%	73.0%	80.0%	96.5%	96.6%	84.8%	86.0%	95.8%	94.8%	0.7%	22.7%
Diversity	0.74	0.75	0.80	0.80	0.75	0.74	0.80	0.80	0.75	0.67	0.83	0.82

The pre-trained network, fine-tuned network, REINVENT, ORGANIC and DrugEx with different G_φ (shown in the parentheses), ε and β were compared.

Subsequently, the fine-tuned network was used as G_φ to be involved in our proposed training method of RL. After the training process converged at 200 epochs (Fig. 3.8B), 10,000 SMILES were generated. Compared to the pre-trained network, there were more unique molecules generated (Table 4.1), most of which were drug-like compounds (Fig. 3.9B and 10A). However, with appropriate ε the fine-tuned network helped the model generate more valid desired SMILES than with the pre-trained network. At the same time the duplication rate was also increased and there were more repetitive molecules being generated. A possible reason is that the percentage of active ligands was higher in the *A2AR* dataset than in the *ZINC* dataset, while the size of *A2AR* dataset was much smaller than *ZINC* dataset, causing a higher number of duplicated samples generated by the fine-tuned model. In addition, a PCA showed that the fine-tuned network was more effective than the pre-trained network as G_φ, as it helped the model in generating molecules with larger diversity and higher similarity to the known active ligands (Fig. 3.9 and 10). These results prove that the exploration strategy is an effective way to assist the model training for generating molecules with similar chemical and biological properties to existing molecules in a specific part of chemical space.

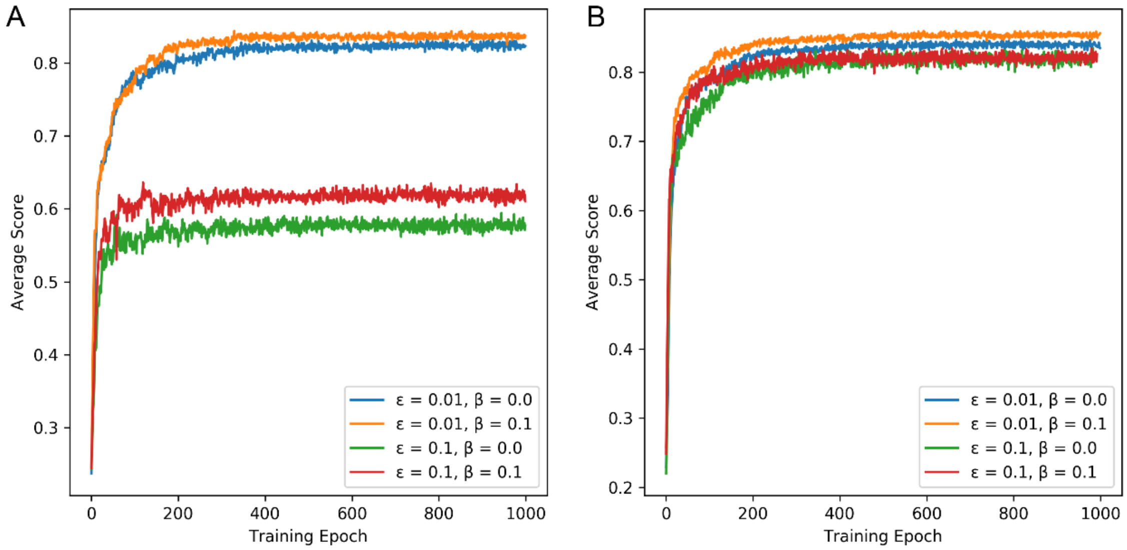


Fig. 3.8: The average score of generated SMILES sequences during the training processes of deep reinforcement learning with different *ε, β and* G_φ. The pre-trained model on *ZINC* dataset (A) and the fine-tuned model on *A2AR* set (B) were used as G_φ. After 200 epochs, the average scores for all training processes converged and all of these models were well trained.

3.3.5. Comparison with other methods

Several papers on SMILES generation using deep learning have been published. Olivecrona *et al.* proposed a method named “REINVENT” [20], in which a new loss function was introduced based on the Bayesian formula for RL,

$$L(\theta) = \left[logP_{Prior}(y_{1:T}) + \sigma R(y_{1:T}) - logP_{Agent}(y_{1:T})\right]^2$$

The authors used all molecules in the ChEMBL database to pre-train an RNN model as the *Priori*. With the parameter σ, they integrated the reward R of each SMILES into the loss function. The final *Agent* model was regarded as the *Posteriori* and trained with the policy gradient. Finally, they successfully identified a large number of active ligands against the dopamine D2 receptor (DRD2).

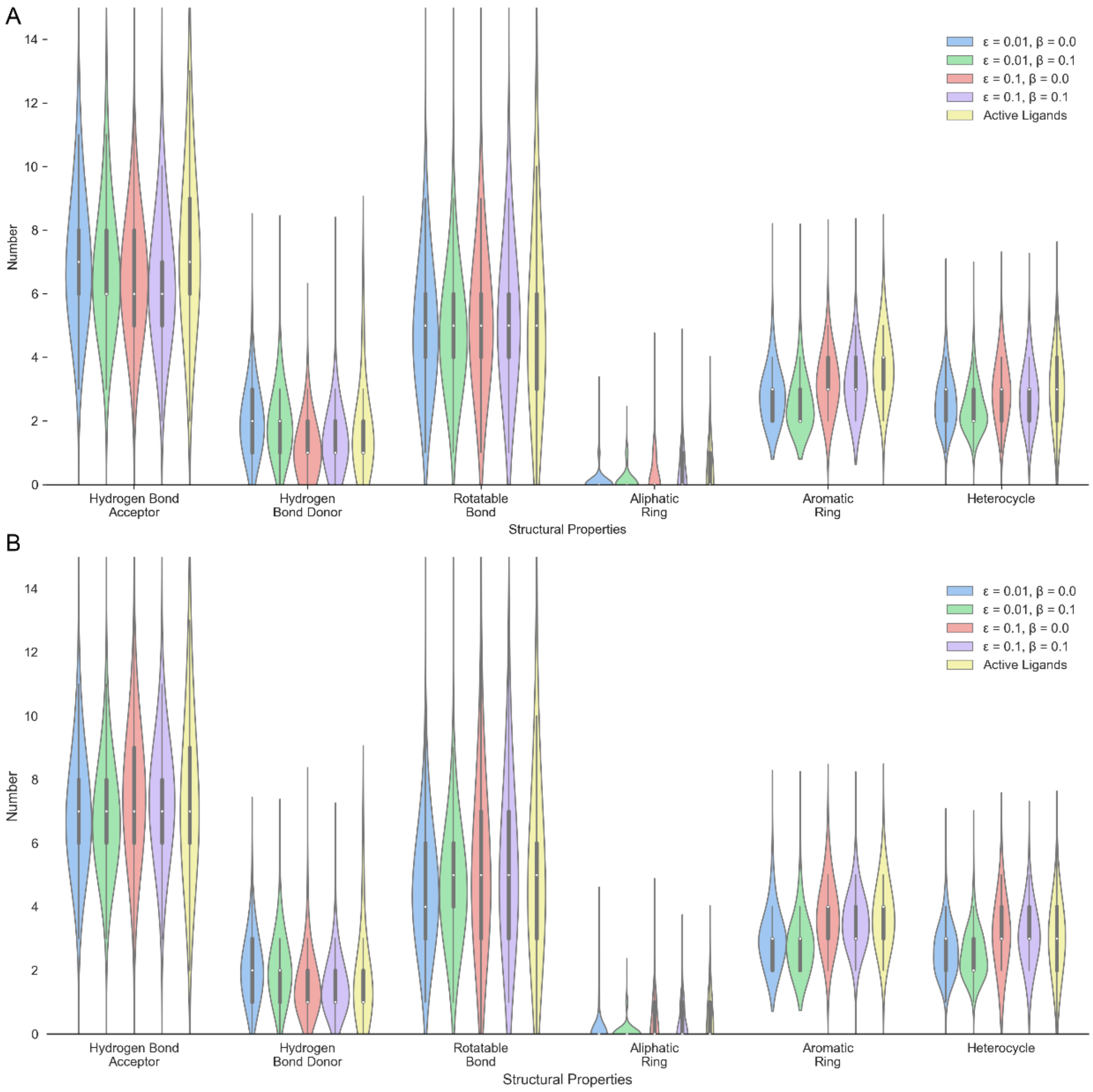


Fig. 3.9: Comparison of the properties of generated molecules by RL models with different ε, β and G_φ. The pre-trained model on *ZINC* dataset (A) and the fine-tuned model on *A2AR* dataset (B) were used as G_φ. These properties included the number of hydrogen bond donors/acceptors, rotatable bonds, aliphatic rings, aromatic rings, and heterocycles.

Likewise, Benjamin *et al.* proposed another method named “ORGANIC” [24] by combining a GAN model for sequence generation and a prediction model to form a comprehensive reward function for RL.

$$R(y_{1:t}) = \lambda R_d(y_{1:T}) + (1-\lambda) R_c(y_{1:T})$$

Here, the reward is represented as the weighted sum of two parts determined by parameter λ: 1) the reward R_c was provided by the prediction model, and 2) the reward R_d was calculated by discriminator neural network D, which was trained with generator simultaneously by minimizing the following loss function:

$$L(\theta) = \sum_{y \in Real} (logD(y_{1:T})) + \sum_{y \in Fake} (log(1 - D(y_{1:T})))$$

With the policy gradient optimization, the final model generated many different desired molecules which were predicted as active ligand against a given target and were similar to the chemical compounds in the ligands dataset. In the following section DrugEx and its performance is compared with these two methods.

The code of REINVENT and ORGANIC was downloaded from GitHub and executed with default parameters ($\sigma = 60$ in REINVENT and $\lambda = 0.5$ in ORGANIC). The prior network of REINVENT and generator network of ORGANIC were initialized with the pre-trained model, and the agent network of REINVENT was started from the fine-tuned model. The RF-based predictor with ECFP6 was exploited as reward function for both methods identical to our own implementation. After these models were trained, 10,000 SMILES sequences were generated for performance comparison with each other (Table 4.1). Our method generated molecules that had the larger diversity at $\varepsilon = 0.1$. While DrugEx did not outperform REINVENT based on the percentage of unique desired SMILES, this value was improved dramatically and closely resembled that of REINVENT when ε was set to 0.01. In addition, although most of the molecules generated by these methods were drug-like molecules (Fig. 3.10A-D), through PCA on PhysChem descriptors (Fig. 3.10E-H) and t-SNE on ECFP6 fingerprints (Figrue 10I-L), we found that molecules generated by our method covered the whole region of chemical space occupied by known active ligands. Conversely, molecules generated by both REINVENT and ORGANIC only covered a small fraction of the desired chemical space and were mostly centered in Rule-of-5 compliant chemical space even though the chemical space for the $A_{2A}R$ transcends this region of space. Moreover, we also used a k-means algorithm to cluster the active ligands in the *A2AR* dataset and generated molecules into 20 groups with the ECFP6 fingerprints of the full compound structure, Murcko scaffold and topological Murcko scaffold. The results indicated that the generated molecules by DrugEx covered all of the clusters that contain active ligands in the *A2AR* dataset, but some of these clusters cannot be covered by REINVENT and ORGANIC (Figure S3). The distribution of the molecules in each cluster generated by DrugEx was closer to active ligands in the *A2AR* dataset than REINVENT and ORGANIC.

Previous work on the binding mechanism between the A_{2A}AR and its ligands identified a number of critical substructures that play an important role to improve binding affinity [36]. For example, the oxygen in the furan ring of ZM241385 and related ligands can form a hydrogen bond with residue N253, the purine ring acts as hydrogen bond donor to N253 and forms π-π interaction with F168 [6]. However, molecules containing such a furan ring tend to be blocking the receptor (antagonists) rather than activating it (agonists). Hence, while the furan ring is common in the set of known A_{2A}R ligands, its presence might not always be favorable for generated ligands. Moreover, in general fused rings have been shown to be important in the chemical structure of drugs [37]. Therefore, we compared the percentage of molecules containing furan rings, fused rings, and benzene rings. Only 0.20% of the desired molecules generated by REINVENT contained a fused ring (Table 4.2) while they were present in 79.09% of active ligands in the A2AR set. Similarly, ORGANIC only generated a very low percentage of molecules containing a fused ring system (0.02%).

With the pre-trained network as G_{φ}, DrugEx produced 9.12% of molecules containing fused rings, while the fine-tuned network improved the percentage of molecules containing fused rings up to 60.69%. Moreover, 95.26% and 99.96% of molecules generated by REINVENT and ORGANIC contained a furan ring, respectively, while this percentage was only 40.29% for known active ligands. In DrugEx, 82.32% of molecules contained a furan ring under the pre-trained network as G_{φ}, similar to the other two methods. However, when the fine-tuned network was used this rate decreased substantially to 66.35%.

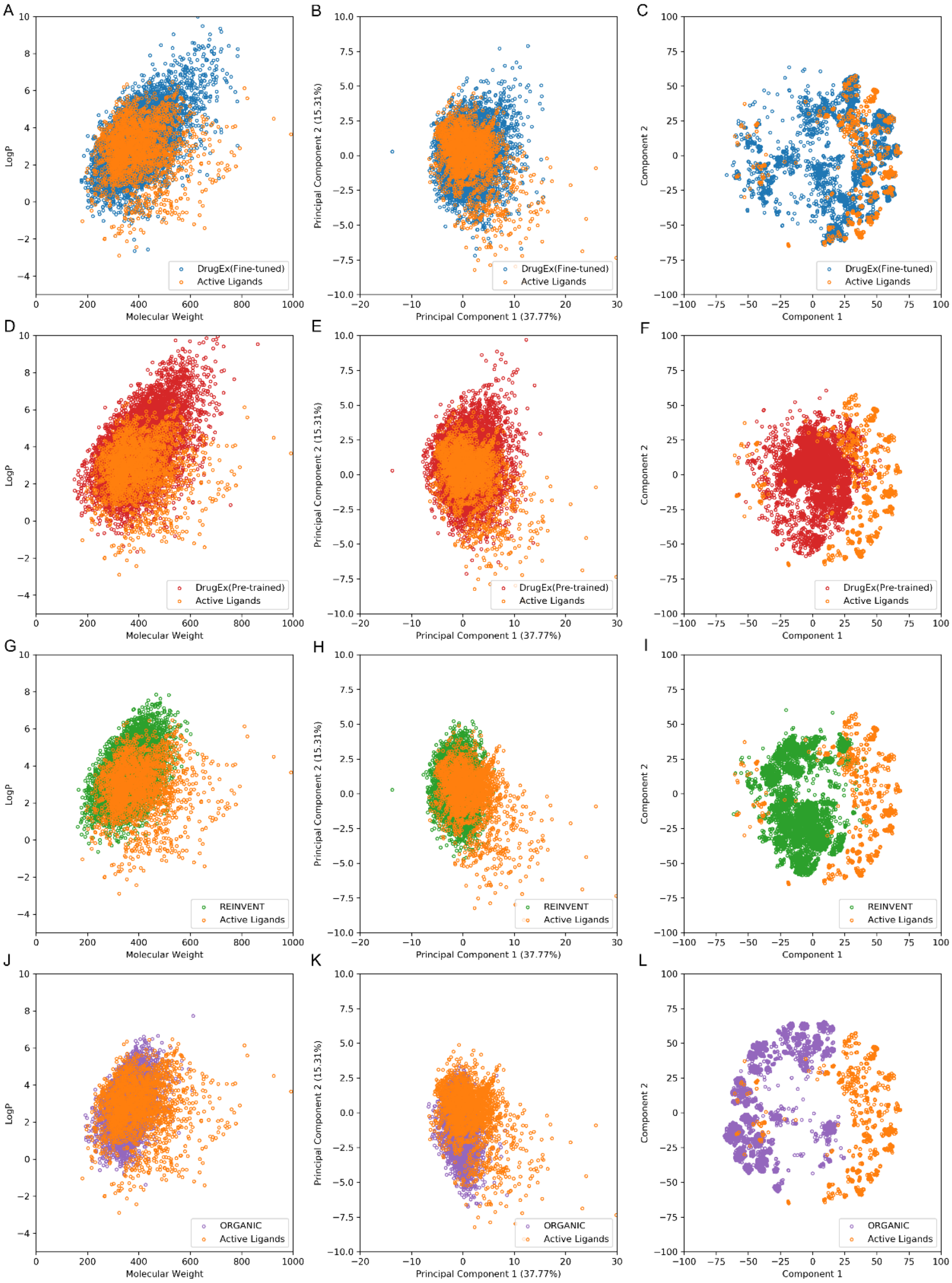


Fig. 3.10: Comparison of the chemical space of active ligands in the *A2AR* set and generated molecules by REINVENT, ORGANIC and DrugEx with different G_φ (shown in parentheses). Chemical Space was represented by logP ~ MW (A, D, G and J) and first two components in PCA on PhysChem descriptors (B, E, H and K) and t-SNE on ECFP6 fingerprints (C, F, I and L).

REINVENT and ORGANIC have been reported to generate various molecules containing different fused ring structures against DRD2 [24,20]. One possible reason they were not able to do so here might lie in the bias of *A2AR* dataset. In table 4.2, we noticed that there were more active ligands containing a furan ring than inactive ligands (four fold difference). This led to both methods only generating molecules containing a furan ring which were prone to be predicted as active. However, both methods neglected to construct more complicated fused rings which is a decisive difference between active and inactive ligands in *A2AR* dataset. These results indicate that DrugEx is more robust to overcome the bias of the training set to generate more similar compounds to known $A_{2A}AR$ ligands (tuned for the target chemical space) and less generic SMILES sequences. Hence, we consider these molecules more appropriate drug candidates against $A_{2A}AR$ than the molecules produced by REINVENT and ORGANIC. As an example, 24 candidate molecules generated by DrugEx were selected and are shown in Fig. 3.11 ordered by the probability score and Tanimoto-distance to the *A2AR* dataset.

Table 4.2: Comparison of the percentage of important substructures contained in the molecules generated by the methods.

		Fused Ring	Furan Ring	Benzene Ring
DrugEx (Pre-trained)		9.12%	82.32%	61.48%
DrugEx (Fine-tuned)		60.69%	66.35%	65.62%
REINVENT		0.20%	95.26%	61.98%
ORGANIC		0.02%	99.96%	39.45%
Pre-trained		24.22%	4.51%	63.31%
Fine-tuned		76.33%	23.82%	72.85%
ZINC		26.66%	3.86%	63.97%
A2AR	**Active**	79.09%	40.29%	75.33%
	Inactive	76.73%	9.33%	70.88%

The table compares DrugEx with pre-trained and fine-tuned model as different G_φ (in the parentheses), REINVENT, ORGANIC, Pre-trained model, Fine-tuned model and the molecules in *ZINC* and *A2AR* dataset.

In REINVENT, the pre-trained model acted as a “priori” in the Bayesian formula to ensure that the generated SMILES are drug-like molecules. The final model was trained by improving the probability of desired generated SMILES while maintaining the probability of undesired generated SMILES similar to the pre-trained model. In DrugEx the pre-trained model was *only* used for initialization and did not directly affect the training process and performance evaluation. The mechanism of DrugEx appears quite similar to a genetic algorithm (GA) previously developed in our group for *de novo* drug design [38]. The exploration strategy can be regarded as “random mutation” in a GA context for sequence generation. Instead of changing the token selection directly, this manipulation just changed the probability distribution of each token in the vocabulary. Furthermore, although “crossover” manipulation was not implemented here, such mutations can still help the model search the unfamiliar chemical space in which the molecules do not have a high probability to be sampled. In contrast to ORGANIC, there was no need to construct another neural network specifically to measure the similarity between generated and real molecules, saving valuable time and resources required to train and select appropriate parameters. Despite the inevitable introduction of some duplicates the molecules generated by DrugEx can be regarded as reasonable drug candidates for A_{2A}AR.

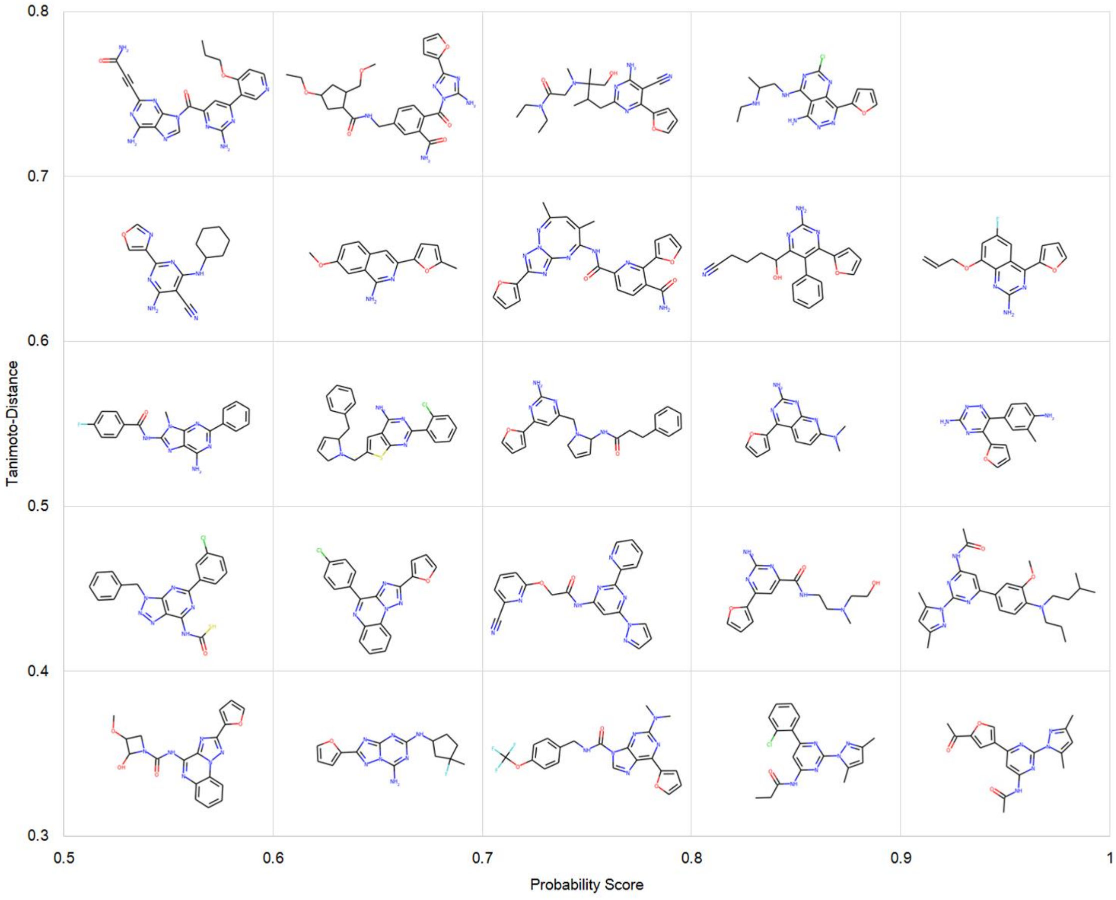


Fig. 3.11: 24 Candidate molecules were selected from 10,000 SMILES sequences generated by DrugEx. These molecules were ordered by the probability score given by the predictor and Tanimoto-distance to *A2AR* dataset.

3.4. Conclusion and future prospect

In this study a new method is proposed to improve the performance of deep reinforcement learning to generate SMILES based ligands for targets of interest. Applied to the $A_{2A}AR$, generated molecules had high diversity combined with chemical and predicted biological properties similar to known active compounds. Previous work has shown that RL cannot guarantee the model to generate molecules distributed over chemical space comparable to ligands of a target of interest. To solve this problem, another pre-trained RNN model was included as exploration strategy to force the model to enlarge the chemical space of the generated molecules during the training process of RL. Compared with other DL-based methods, DrugEx generated molecules with larger diversity and higher similarity to known active ligands, albeit at the expense of more inactive or duplicated molecules.

In future work, the aim is to update DrugEx with multi-objective optimization. As a given drug (candidate) likely binds some other targets (*i.e.* off-target efficacy) which can cause side-effects [39]. Incorporating multiple objectives in SMILES generation will allows the search for ways to eliminate potential off-target affinity.

Declarations

Availability of data and materials

The data used in this study is publicly available ChEMBL data, the algorithm published in this manuscript is made available at https://github.com/XuhanLiu/DrugEx.

Authors' Contributions

XL and GJPvW conceived the study and performed the experimental work and analysis. KY, APIJ and HWTvV provided feedback and critical input. All authors read, commented on and approved the final manuscript.

Acknowledgements

XL thanks Chinese Scholarship Council (CSC) for funding, GJPvW thanks the Dutch Research Council and Stichting Technologie Wetenschappen (STW) for financial support (STW-Veni #14410).

Competing Interests

The authors declare that they have no competing interests

References

1. Lv X, Liu J, Shi Q, Tan Q, Wu D, Skinner JJ, Walker AL, Zhao L, Gu X, Chen N, Xue L, Si P, Zhang L, Wang Z, Katritch V, Liu ZJ, Stevens RC (2016) In vitro expression and analysis of the 826 human G protein-coupled receptors. Protein & cell 7 (5):325-337. doi:10.1007/s13238-016-0263-8
2. Dorsam RT, Gutkind JS (2007) G-protein-coupled receptors and cancer. Nature reviews Cancer 7 (2):79-94. doi:10.1038/nrc2069
3. Hauser AS, Attwood MM, Rask-Andersen M, Schioth HB, Gloriam DE (2017) Trends in GPCR drug discovery: new agents, targets and indications. Nature reviews Drug discovery 16 (12):829-842. doi:10.1038/nrd.2017.178
4. Chen JF, Eltzschig HK, Fredholm BB (2013) Adenosine receptors as drug targets--what are the challenges? Nature reviews Drug discovery 12 (4):265-286. doi:10.1038/nrd3955
5. Liu W, Chun E, Thompson AA, Chubukov P, Xu F, Katritch V, Han GW, Roth CB, Heitman LH, AP IJ, Cherezov V, Stevens RC (2012) Structural basis for allosteric regulation of GPCRs by sodium ions. Science 337 (6091):232-236. doi:10.1126/science.1219218
6. Jaakola VP, Griffith MT, Hanson MA, Cherezov V, Chien EY, Lane JR, Ijzerman AP, Stevens RC (2008) The 2.6 angstrom crystal structure of a human A2A adenosine receptor bound to an antagonist. Science 322 (5905):1211-1217. doi:10.1126/science.1164772
7. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, Light Y, McGlinchey S, Michalovich D, Al-Lazikani B, Overington JP (2012) ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic acids research 40 (Database issue):D1100-1107. doi:10.1093/nar/gkr777
8. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521 (7553):436-444. doi:10.1038/nature14539
9. Mamoshina P, Vieira A, Putin E, Zhavoronkov A (2016) Applications of Deep Learning in Biomedicine. Molecular pharmaceutics 13 (5):1445-1454. doi:10.1021/acs.molpharmaceut.5b00982
10. Miotto R, Wang F, Wang S, Jiang X, Dudley JT (2017) Deep learning for healthcare: review, opportunities and challenges. Briefings in bioinformatics. doi:10.1093/bib/bbx044
11. Cherkasov A, Muratov EN, Fourches D, Varnek A, Baskin, II, Cronin M, Dearden J, Gramatica P, Martin YC, Todeschini R, Consonni V, Kuz'min VE, Cramer R, Benigni R, Yang C, Rathman J, Terfloth L, Gasteiger J, Richard A, Tropsha A (2014) QSAR modeling: where have you been? Where are you going to? Journal of medicinal chemistry 57 (12):4977-5010. doi:10.1021/jm4004285
12. Ekins S (2016) The Next Era: Deep Learning in Pharmaceutical Research. Pharmaceutical research 33 (11):2594-2603. doi:10.1007/s11095-016-2029-7
13. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T (2018) The rise of deep learning in drug discovery. Drug discovery today. doi:10.1016/j.drudis.2018.01.039
14. Rogers D, Hahn M (2010) Extended-connectivity fingerprints. Journal of chemical information and modeling 50 (5):742-754. doi:10.1021/ci100050t
15. Lenselink EB, Ten Dijke N, Bongers B, Papadatos G, van Vlijmen HWT, Kowalczyk W, AP IJ, van Westen GJP (2017) Beyond the hype: deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set. Journal of cheminformatics 9 (1):45. doi:10.1186/s13321-017-0232-0
16. Jaeger S, Fulle S, Turk S (2018) Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition. Journal of chemical information and modeling 58 (1):27-35. doi:10.1021/acs.jcim.7b00616
17. Kadurin A, Nikolenko S, Khrabrov K, Aliper A, Zhavoronkov A (2017) druGAN: An Advanced Generative Adversarial Autoencoder Model for de Novo Generation of New Molecules with Desired Molecular Properties in Silico. Molecular pharmaceutics 14 (9):3098-3104.

doi:10.1021/acs.molpharmaceut.7b00346

18. Duvenaud D, Maclaurin D, Aguilera-Iparraguirre J, Gómez-Bombarelli R, Hirzel T, Aspuru-Guzik A, Adams RP (2015) Convolutional Networks on Graphs for Learning Molecular Fingerprints. ArXiv:1509.09292
19. Gupta A, Muller AT, Huisman BJH, Fuchs JA, Schneider P, Schneider G (2018) Generative Recurrent Networks for De Novo Drug Design. Molecular informatics 37 (1-2). doi:10.1002/minf.201700111
20. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. Journal of cheminformatics 9 (1):48. doi:10.1186/s13321-017-0235-x
21. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529 (7587):484-489. doi:10.1038/nature16961
22. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative Adversarial Networks. ArXiv:1406.2661
23. Yu L, Zhang W, Wang J, Yu Y (2016) SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. ArXiv:1609.05473
24. Benjamin S-L, Carlos O, Gabriel L. G, Alan A-G (2017) Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). doi:10.26434/chemrxiv.5309668.v3
25. Preuer K, Renz P, Unterthiner T, Hochreiter S, Klambauer GUN (2018) Frechet ChemNet Distance: A metric for generative models for molecules in drug discovery. Journal of chemical information and modeling. doi:10.1021/acs.jcim.8b00234
26. Benhenda M (2017) ChemGAN challenge for drug discovery: can AI reproduce natural chemical diversity? ArXiv:1708.08227
27. Schneider G, Fechner U (2005) Computer-based de novo design of drug-like molecules. Nature reviews Drug discovery 4 (8):649-663. doi:10.1038/nrd1799
28. Sterling T, Irwin JJ (2015) ZINC 15--Ligand Discovery for Everyone. Journal of chemical information and modeling 55 (11):2324-2337. doi:10.1021/acs.jcim.5b00559
29. Gaulton A, Hersey A, Nowotka M, Bento AP, Chambers J, Mendez D, Mutowo P, Atkinson F, Bellis LJ, Cibrian-Uhalte E, Davies M, Dedman N, Karlsson A, Magarinos MP, Overington JP, Papadatos G, Smit I, Leach AR (2017) The ChEMBL database in 2017. Nucleic acids research 45 (D1):D945-D954. doi:10.1093/nar/gkw1074
30. RDKit: Open-Source Cheminformatics Software. http://www.rdkit.org.
31. Reddy AS, Zhang S (2013) Polypharmacology: drug discovery for the future. Expert review of clinical pharmacology 6 (1):41-47. doi:10.1586/ecp.12.74
32. PyTorch. https://pytorch.org/.
33. Kingma DP, Ba J (2014) Adam: A Method for Stochastic Optimization. arXiv:1412.6980
34. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv:1412.3555
35. Segler MHS, Kogej T, Tyrchan C, Waller MP (2018) Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. Acs Central Sci 4 (1):120-131. doi:10.1021/acscentsci.7b00512
36. Jaakola VP, Lane JR, Lin JY, Katritch V, Ijzerman AP, Stevens RC (2010) Ligand binding and subtype selectivity of the human A(2A) adenosine receptor: identification and characterization of essential amino acid residues. The Journal of biological chemistry 285 (17):13032-13044. doi:10.1074/jbc.M109.096974

37. Feher M, Schmidt JM (2003) Property distributions: differences between drugs, natural products, and molecules from combinatorial chemistry. Journal of chemical information and computer sciences 43 (1):218-227. doi:10.1021/ci0200467
38. Lameijer EW, Kok JN, Back T, Ijzerman AP (2006) The molecule evoluator. An interactive evolutionary algorithm for the design of drug-like molecules. Journal of chemical information and modeling 46 (2):545-552. doi:10.1021/ci050369d
39. Giacomini KM, Krauss RM, Roden DM, Eichelbaum M, Hayden MR, Nakamura Y (2007) When good drugs go bad. Nature 446 (7139):975-977. doi:10.1038/446975a

Table S3.1: All tokens in vocabulary for SMILES sequence construction with RNN model.

Atoms				**Bonds**	**Controls**		
Common Atoms			**Aromatic Atoms**	**--**	**Rings**	**Branchs**	**On-Off**
B	[B-]	[O-]	[cH-]	-	1	(	GO
C	[BH-]	[O]	[n+]	=	2	)	EOS
F	[C+]	[P+]	[nH]	#	3		
I	[C-]	[PH]	[s+]		4		
Cl	[CH-]	[Br+]	c		5		
N	[CH]	[S+]	n		6		
O	[C]	[SH]	o		7		
P	[N+]	[SiH3]	p		8		
Br	[NH+]	[SiH]	s		9		
S	[N]	[Si]					
		[Sn]					

Considering that there are no drug-like molecules containing more than 10 rings, we omitted the token “0” and “%” for the construction of more than 10 rings. In addition, we ignored the isomerism of molecules and ionic bond, therefore we removed the “@”, ‘\’, ‘/’, ‘.’ and all metal ion.

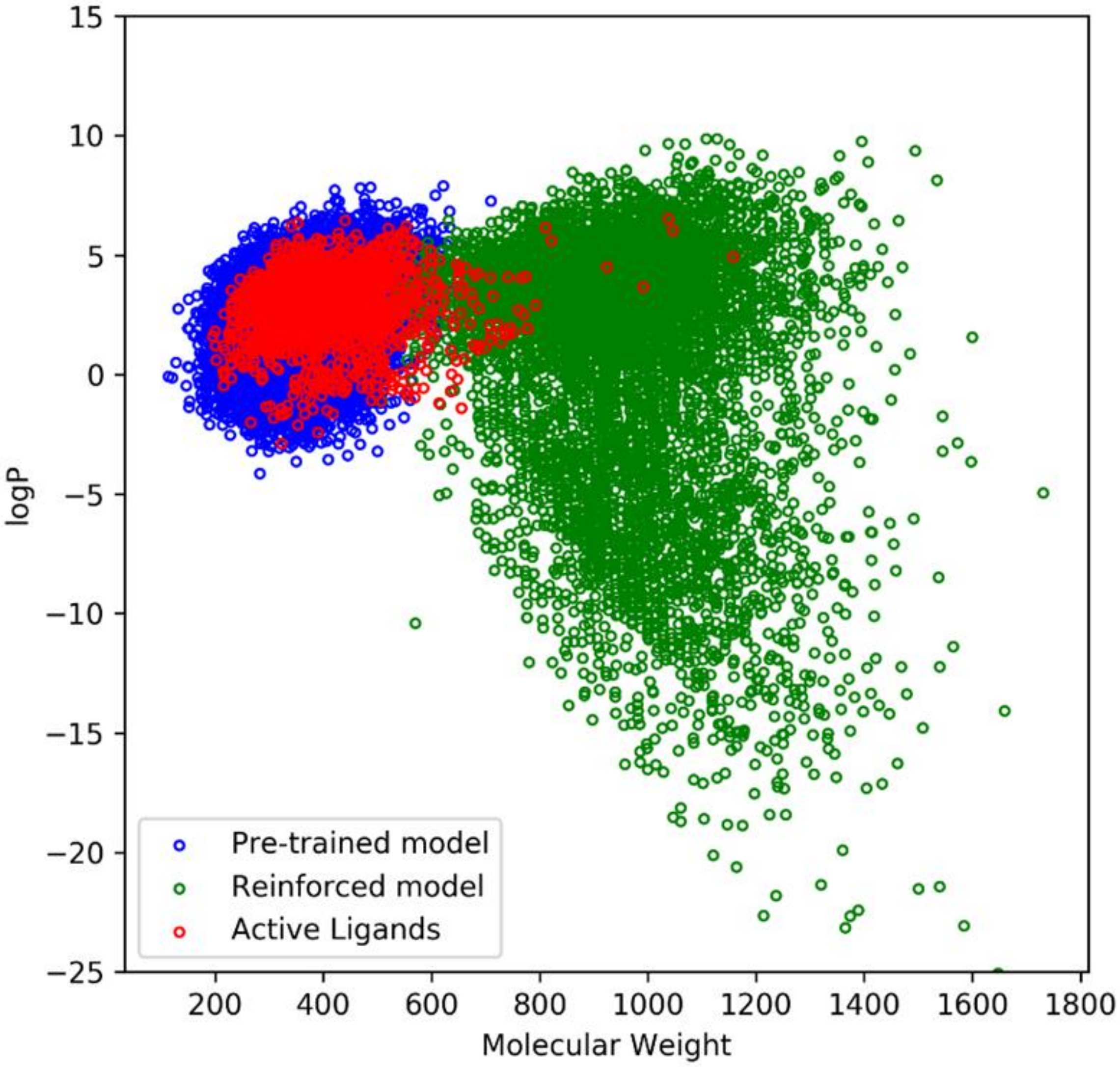
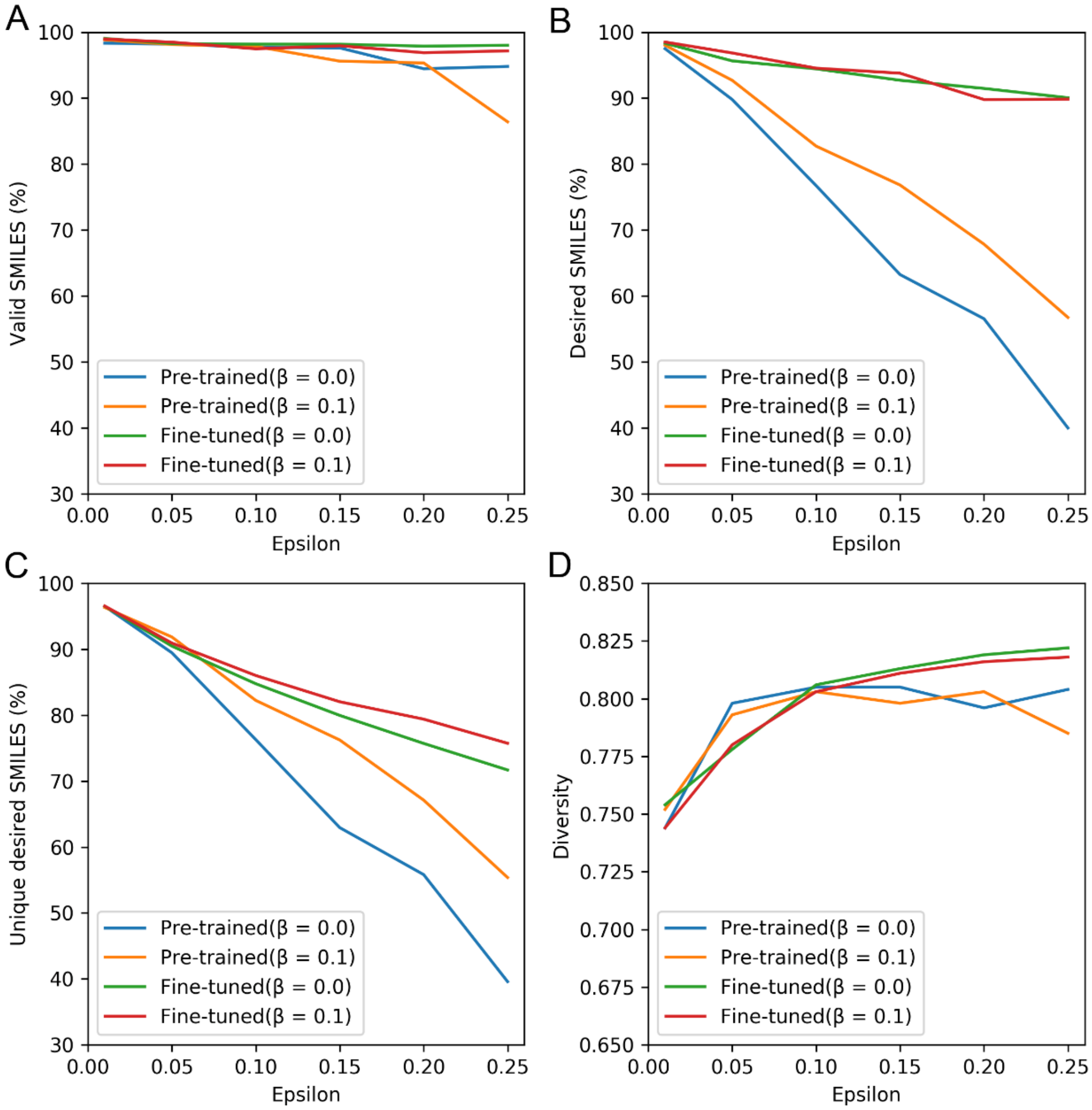


Figure S3.1: The chemical space of generated molecules by pre-trained models, traditional reinforced model and active ligands in the *A2AR* set. The chemical space was represented as logP ~ MW. The generated molecules by pre-trained model covered the greater part of space of known active ligands, while the molecules generated by reinforced model were distributed in a distinct region, which cannot be regarded drug-like although the compounds were predicted as active ligands.



TableS3.2: The performance of DrugEx with different G_φ (pre-trained and fine-tuned model) and hyperparameters (including *ε* and *β*). These performance included the percentage of valid SMILES (A), desired SMILES (B) and unique desired SMIIES (C) and diversity (D).

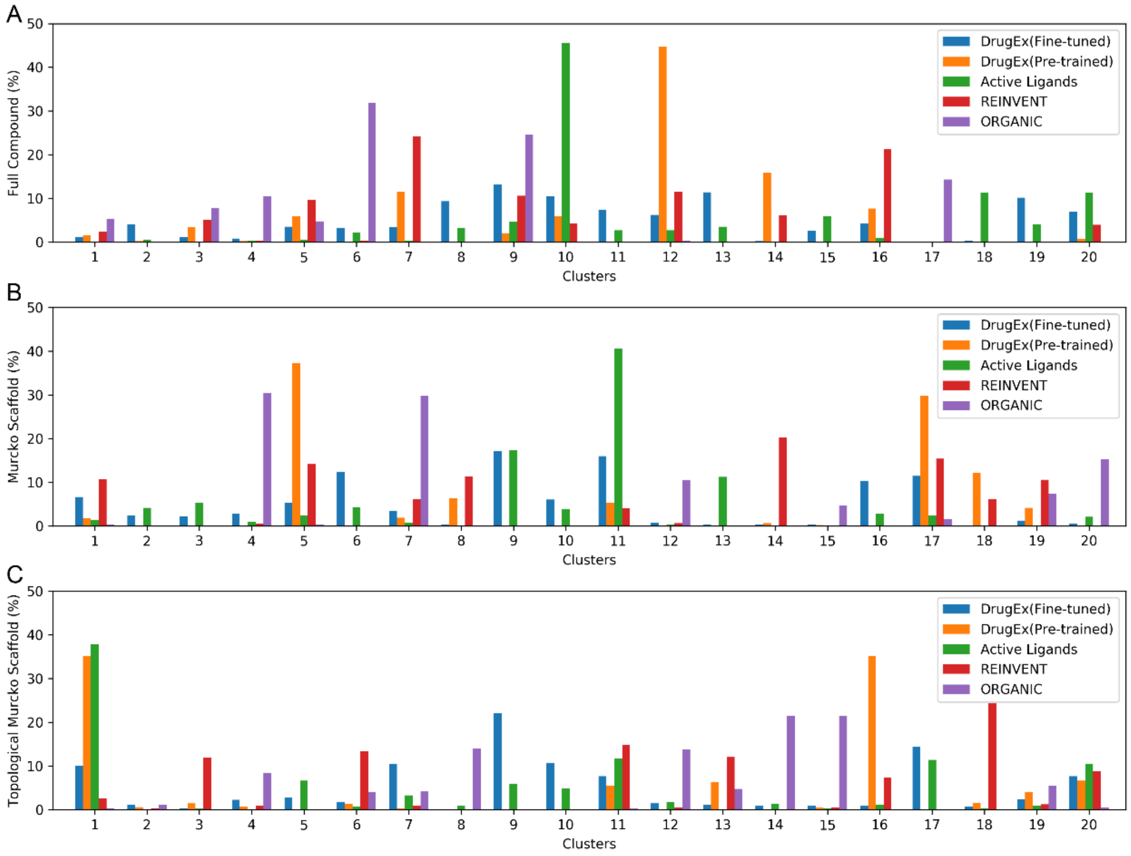


Figure S3.3: The percentage of molecules in 20 groups clustered by k-means algorithm on ECFP6 fingerprints of generated molecules with full compound (A), Murcko scaffold (B) and topological Murcko scaffold (C). These molecules included active ligands in *A2AR* dataset and molecules generated by REINVENT, ORGANIC and DrugEx with different G_φ (shown in the parentheses)

Chapter 4

DrugEx v2: *de novo* design of drug molecules by Pareto-based multi-objective reinforcement learning in polypharmacology

Xuhan Liu, Kai Ye, Herman W. T. van Vlijmen, Michael T. M. Emmerich, Adriaan P. IJzerman, Gerard J. P. van Westen*. Journal of Cheminformatics (2021). Https://doi.org/10.1186/s13321-021-00561-9

Abstract

In polypharmacology, ideal drugs are required to bind to multiple specific targets to enhance efficacy or to reduce resistance formation. Although deep learning has achieved a breakthrough in *de novo* drug design, most of its applications only focus on a single drug target to generate drug-like active molecules in spite of the reality that drug molecules often interact with more than one target which can have desired (polypharmacology) or undesired (toxicity) effects. In a previous study we proposed a new method named *DrugEx* that integrates an exploration strategy into RNN-based reinforcement learning to improve the diversity of the generated molecules. Here, we extended our *DrugEx* algorithm with multi-objective optimization to generate drug molecules towards more than one specific target (two adenosine receptors, A_1AR and A_{2A}AR, and the potassium ion channel hERG in this study). In our model, we applied an RNN as the *agent* and machine learning predictors as the *environment*, both of which were pre-trained in advance and then interplayed under the reinforcement learning framework. The concept of evolutionary algorithms was merged into our method such that *crossover* and *mutation* operations were implemented by the same deep learning model as the *agent*. During the training loop, the agent generates a batch of SMILES-based molecules. Subsequently scores for all objectives provided by the *environment* are used to construct Pareto ranks of the generated molecules with non-dominated sorting and Tanimoto-based crowding distance algorithms. Here, we adopted GPU acceleration to speed up the process of Pareto optimization. The final reward of each molecule is calculated based on the Pareto ranking with the ranking selection algorithm. The agent is trained under the guidance of the reward to make sure it can generate more desired molecules after convergence of the training process. All in all we demonstrate generation of compounds with a diverse predicted selectivity profile towards multiple targets, offering the potential of high efficacy and low toxicity.

Keywords: deep learning; adenosine receptors; cheminformatics; reinforcement learning; multi-objective optimization; exploration strategy.

4.1. Introduction

The ‘one drug, one target, one disease’ paradigm, which has dominated the field of drug discovery for many years, has made great contributions to drug development and the understanding of their molecular mechanisms of action [1]. However, this strategy is encountering problems due to the intrinsic promiscuity of drug molecules, *i.e.* recent studies showed that one drug molecule could interact with six protein targets on average [2]. Side effects of drugs caused by binding to unexpected off-targets are one of the main reasons of clinical failure of drug candidates and even withdrawal of FDA-approved novel drugs [3,4]. Up to now, more than 500 drugs have been withdrawn from the market due to fatal toxicity [5]. Yet, disease often results from the perturbation of biological systems by multiple genetic and/or environmental factors, thus complex diseases are more likely to require treatment through modulating multiple targets simultaneously. Therefore, it is crucial to shift the drug discovery paradigm to “polypharmacology” for many complex diseases [6,7].

In polypharmacology, ideal drugs are required to bind to multiple specific targets to enhance efficacy or to reduce resistance formation (in which case multiple targets can be multiple mutants of a single target) [8]. It has been shown that partial inhibition of a small number of targets can be more efficient than the complete inhibition of a single target, especially for complex and multifactorial diseases [6,9]. In parallel, common structural and functional similarity of proteins results in drugs binding to off-targets. Hence we also demand drugs to have a high target selectivity to avoid binding to unwanted target proteins. For example, the adenosine receptors (ARs) are a class of rhodopsin-like G protein-coupled receptors (GPCRs) having adenosine as the endogenous ligand. Adenosine and ARs are ubiquitously distributed throughout the human tissues, and their interactions trigger a wide spectrum of physiological and pathological functions. There are four subtypes of ARs, including A_1, A_{2A}, A_{2B} and A_3, each of which has a unique pharmacological profile, tissue distribution, and effector coupling [10,11]. The complexity of adenosine signaling and the widespread distribution of ARs have always given rise to challenges in developing target-specific drugs [12]. In addition to the similarity of the pharmacophores of some generic proteins (*e.g.* the human Ether-à-go-go-Related Gene, hERG) should also be taken into consideration as they can be sensitive to binding exogenous ligands and cause side effects. hERG is the alpha subunit of a potassium ion channel [13] and has an inclination to interact with drug molecules because of its larger inner vestibule as the ligand binding pocket [14]. When hERG is inhibited this may cause long QT syndrome [15].

In addition to visual recognition, natural language processing and decision making, deep learning has been increasingly applied in drug discovery [16]. It does not only perform well in prediction models for virtual screening, but is also used to construct generative models for drug *de novo* design and/or drug optimization [17]. For example, our group implemented a fully-connected deep neural network (DNN) to construct a proteochemometric model (PCM) with all high quality ChEMBL data [18] for prediction of ligand bioactivity [19]. Its performance was shown to be better than other shallow machine learning methods. Moreover, we also developed a generative model with recurrent neural networks (RNNs), named *DrugEx* for SMILES-based *de novo* drug design [20]. It was shown that the generated molecules had large diversity and were similar to known ligands to some extent to make sure that reliable and diverse drug candidates can be designed.

Since the first version of *DrugEx* (*v1*) demonstrated effectiveness for designing novel A_{2A}AR ligands, we began to extend this method for drug design toward multiple targets. In this study, we updated *DrugEx* to the second version (*v2*) through merging crossover and mutation operations, which were derived from evolutionary algorithms, into the reinforcement learning (RL) framework. We also used Pareto ranking for multi-objective selection. In order to evaluate the performance of our additions we tested our method into both multi-target and target-specific cases. For the multi-target case, desired molecules should have a high affinity towards both A_1AR and A_{2A}AR. In the target-specific case, on the other hand, we required molecules to have only high affinity towards the A_{2A}AR but a low affinity to the A_1AR. In order to decrease toxicity and adverse events, molecules were additionally obliged to have a low affinity for hERG in both cases. It is worth noting that generated molecules should also be chemically diverse and have similar physico-chemical properties to known ligands. All python code for this study is freely available at http://github.com/XuhanLiu/DrugEx.

4.2. Materials and methods

4.2.1. Data source

Drug like molecules represented as SMILES format were downloaded from the ChEMBL database (version 26). After data preprocessing, including recombining charges, removing metals and small fragments, we collected 1.7 million molecules and named it the *ChEMBL* set, used for SMILES syntax learning. This data preprocessing step was implemented in RDKit [21]. Furthermore, 25,731 ligands were extracted from the ChEMBL database to construct the *LIGAND* set, which had bioactivity measurements towards the human A_1AR, A_{2A}AR, and hERG. The *LIGAND* set was used to construct prediction models for each target and fine-tuning the generative models. The number of ligands and bioactivities for these three targets in the *LIGAND* set is represented in Table 4.1. Duplicate items were removed and if multiple measurements for the same ligands existed, the average pChEMBL value (pX, including pKi, pKd, pIC50, or pEC50) was calculated. To judge if a molecule is active or not, we defined the threshold of bioactivity as pX = 6.5. If the pX < 6.5, the compound was predicted as undesired (low affinity to the given target); otherwise, it was regarded as desired (having high affinity) [19].

4.2.2. Prediction model

In order to predict the pX for each generated molecule for a given target, regression QSAR models were constructed with different machine learning algorithms. To increase the chemical diversity available for the QSAR model we included lower quality data without pChEMBL value, *i.e.* molecules that were labeled as "Not Active" or without a defined pX value. For these data points we defined a pX value of 3.99 (slightly smaller than 4.0) to eliminate the imbalance of the dataset and guarantee the model being able to predict the negative samples. During the training process, sample weights for low quality data were set as 0.1, while the data with exact pX were set as 1.0. This allowed us to particularly incorporate the chemical diversity, while avoiding degradation of model quality. Descriptors used as input were ECFP6 fingerprints [22] with 2048 bits (2048 dimensions, or 2048D) calculated by the RDKit Morgan Fingerprint algorithm (using a three-bond radius). Moreover, the following 19D physico-chemical descriptors were used: molecular weight, logP, number of H bond acceptors and donors, number of rotatable bonds, number of amide bonds, number of bridge head atoms, number of hetero atoms, number of spiro atoms, number of heavy atoms, the fraction of SP3 hybridized carbon atoms, number of aliphatic rings, number of saturated rings, number of total rings, number of aromatic rings, number of heterocycles, number of valence electrons, polar surface area and Wildman-Crippen MR value. Hence, each molecule in the dataset was transformed into a 2067D vector. Before being input into the model, the value of input vectors were normalized to the range of [0, 1] by the MinMax method. Model output value is the probability whether a given chemical compound was active based on this vector.

Table 4.1: The number of ligands and bioactivities for each of the human protein targets A_1AR, A_{2A}AR and hERG in the *LIGAND* set.

	A_1AR	**A_{2A}AR**	**hERG**
Total Ligands	7,700	8,406	16,733
Bioactivities	13,100	12,129	22,156
Active Ligands (pX >= 6.5)	1,990	2,511	924
Inactive Ligands (pX < 6.5)	1,859	1,709	6,438
Inactive Ligands (No pX)	1,764	1,993	1,275
Other Ligands	2,087	4,704	8,906

Four algorithms were benchmarked for QSAR model construction, Random Forest (RF), Support Vector Machine (SVM), Partial Least Squares regression (PLS), and Multi-task Deep Neural Network (MT-DNN). RF, SVM and PLS models were implemented through Scikit-Learn [23], and the MT-DNN model through PyTorch [24]. In the RF, the number of trees was set as 1000 and split criterion was “gini”. In the SVM, a radial basis function

(RBF) kernel was used and the parameter space of C and γ were set as [2^{-5}, 2^{15}] and [2^{-15}, 2^{5}], respectively. In the MT-DNN, the architecture contained three hidden layers activated by a rectified linear unit (ReLU) between input and output layers, and the number of neurons were 2048, 4000, 2000, 1000 and 3 in these subsequent layers. The training process consisted of 100 epochs with 20% of hidden neurons randomly dropped out between each layer. The mean squared error was used to construct the loss function and was optimized by the Adam algorithm [25] with a learning rate of 10^{-3}.

4.2.3. Generative model

As in *DrugEx v1*, we organized the vocabulary for the SMILES construction. Each SMILES-format molecule in the *ChEMBL* and *LIGAND* sets was split into a series of tokens. Then all tokens existing in this dataset were collected to construct the SMILES vocabulary. The final vocabulary contained 84 tokens (Table S4.1) which were selected and arranged sequentially into valid SMILES sequences through correct grammar.

The RNN model constructed for sequence generation contained six layers: one input layer, one embedding layer, three recurrent layers and one output layer. After being represented by a sequence of tokens, molecules can be received as categorical features by the input layer. In the embedding layer, vocabulary size, and embedding dimension were set to 84 and 128, meaning each token could be transformed into a 128 dimensional vector. For a recurrent layer, the long-short term memory (LSTM) was used as recurrent cell with 512 hidden neurons instead of the gated recurrent unit (GRU) [26] which was employed only in *DrugEx v1*. The output at each position was the probability that determined which token in the vocabulary would be chosen to grow the SMILES string.

During the training process we put a start token (GO) at the beginning of a batch of data as input and an end token (END) at the end of the same batch of data as output. This ensures that our generative network could choose correct tokens each time based on the sequence it had generated previously. A negative log likelihood function was used to construct the loss function to guarantee that the token in the output sequence had the largest probability to be chosen after being trained. In order to optimize the parameters of the model, the Adam algorithm [25] was used for the optimization of the loss function. Here, the learning rate was set at 10^{-3}, the batch size was 512, and training steps were set to 1000 epochs.

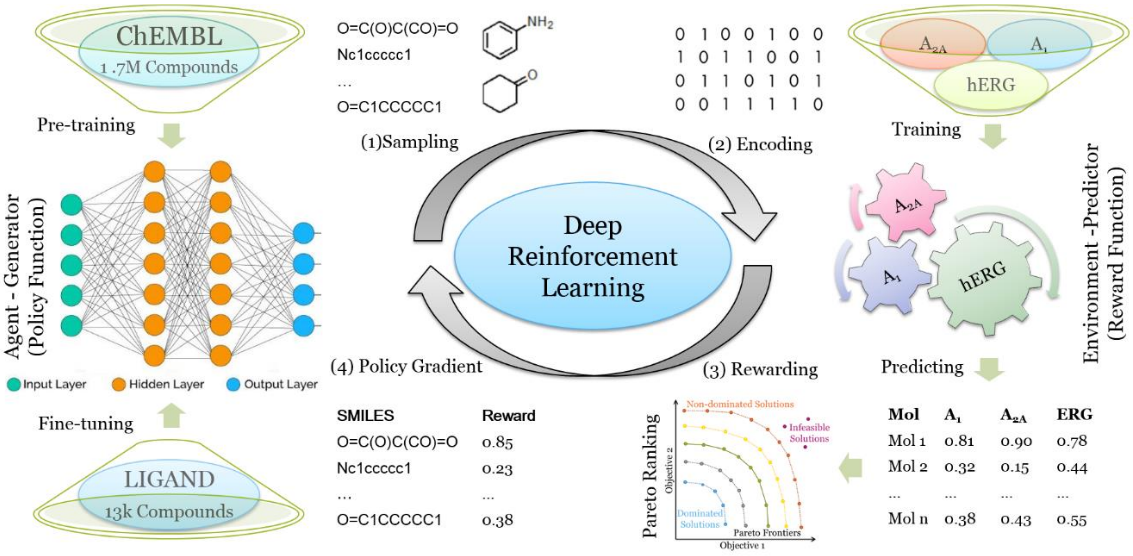


Fig. 4.1: **The workflow of the training process of our deep learning-based molecule generator *DrugEx2* utilizing reinforcement learning.** After the generator has been pre-trained/fine-tuned, (1) a batch of SMILES are generated by sampling tokens step by step based on the probability calculated by the generator; (2) These valid SMILES are parsed to be molecules and encoded into descriptors to get the predicted pXs with well-trained predictors; (3) The predicted pXs are transformed into a single value as the reward for each molecule based on Pareto optimization; (4) These SMILES sequences and their rewards are sent back to the generator for training with policy gradient methods. These four steps constitute the training loop of reinforcement learning.

4.2.4. Reinforcement learning

SMILES sequence construction under the RL framework can be viewed as a series of decision-making steps (Fig. 4.1). The generator (G) and the predictors (Q) are regarded as the policy and reward function, respectively. In this study we used multi-objective optimization (MOO), and each objective was a requirement to be achieved maximally for each scenario, albeit with differences in desirability. Our aim was defined by the following problem statement:

$$maximize\ R_1, \quad maximize\ R_2, \quad \ldots, \quad maximize\ R_n$$

Here, n equals the number of objectives (n = 3 in this study), and R_i, the score for each objective i, was calculated as follows:

$$R_i = \begin{cases} minmax(pX_i), & if\ high\ affinity\ required \\ 1 - minmax(pX_i), & if\ low\ affinity\ required \\ 0, & if\ SMILES\ invalid \end{cases}$$

here the pX_i (the range from 3.0 to 10.0) was the prediction score given by each predictor for the i^{th} target, which was normalized to the interval [0, 1] as the reward score. If having no or low affinity for a target was required (off-target) this score would be subtracted from 1 (inverting it). For the multi-target case, the objective function is:

$$\begin{cases} R_{A1} = minmax(pX_{A1}) \\ R_{A2A} = minmax(pX_{A2A}) \\ R_{hERG} = 1 - minmax(pX_{hERG}) \end{cases}$$

while the objective function for the target-specific case, is:

$$\begin{cases} R_{A1} = 1 - minmax(pX_{A1}) \\ R_{A2A} = minmax(pX_{A2A}) \\ R_{hERG} = 1 - minmax(pX_{hERG}) \end{cases}$$

In order to evaluate the performance of the generators, three coefficients are calculated with the generated molecules, including validity, desirability, and uniqueness which are defined as:

$$\text{Validity} = \frac{N_{valid}}{N_{total}}$$

$$\text{Desirability} = \frac{N_{desired}}{N_{total}}$$

$$\text{Uniqueness} = \frac{N_{unique}}{N_{total}}$$

where N_{total} is the total number of molecules, N_{valid} is the number of the molecules parsed by the valid SMILES sequences, N_{unique} is the number of molecules which are different from others in the dataset, and $N_{desired}$ is the number of desired molecules. Here, we determine whether generated molecules are desired based on the reward R_i if all of them are larger than the threshold (0.5 by default when pX = 6.5). In addition, we calculated the SA score (from 1 to 10) for each molecule to measure the synthesizability of which larger value means more difficult to be synthesized [27]. And we also computed the QED (from 0 to 1) score to evaluate the drug-likeness of which larger value means more drug-like for each molecule [28]. The calculation of both SA and QED scores were implemented by RDKit.

To orchestrate and combine these different objectives, we compared two different reward schemes: the Pareto front (PF) scheme and the weighted sum (WS) scheme. These were defined as follows:

(a) Weighted sum (WS) scheme: the weight for each function is not fixed but dynamic, and depends on the desired ratio for each objective, which is defined as:

$$\mathrm{r}_i = \frac{N_i^s}{N_i^l}$$

here for objective i the N_i^s and N_i^l are the number of generated molecules which have a score smaller or larger than the threshold. Moreover, the weight is normalized ratio defined as:

$$w_i = \frac{r_i}{\sum_{k=1}^{M} r_k}$$

and the final reward R^* was calculated by

$$R^* = \sum_{i=1}^{n} w_i R_i \, ,$$

(b) Pareto front (PF) scheme: operates on the desirability score, which is defined as

$$\mathrm{D}_i = \begin{cases} 1, & if\ R_i > t_i \\ R_i/t_i\,, & if\ R_i \le t_i \end{cases}$$

where t_i is the threshold of the i^{th} objective, and we set all of objectives had the same threshold as 0.5 as stated in the methods. Given two solutions m_1 and m_2 with their scores ($x_1, x_2, ..., x_n$) and ($y_1, y_2, ..., y_n$), then m_1 is said to Pareto dominate m_2 if and only if:

$$\forall\ \mathrm{j} \in \{1, \dots, \mathrm{n}\}: x_j \ \ge y_j\ and\ \exists\ \mathrm{j} \in \{1, \dots, \mathrm{n}\}: x_j > y_j$$

otherwise, m_1 and m_2 are non-dominated with each other. After the dominance between all pair of solutions being determined, the non-dominated scoring algorithm [29] is exploited to obtain different layers of Pareto frontiers which consist of a set of solutions. The solutions in the top layer are dominated by the other solutions in the lower layer [30]. In order to speed up the non-dominated sorting algorithm, we employed *PyTorch* to implement this procedure with GPU acceleration. After obtaining the frontiers ranking from dominated solutions to dominant solutions, the molecules were ranked based on the average of Tanimoto-distance instead of crowding distance with other molecules in the same frontier, and molecules with larger distances were ranked on the top. The final reward R^* is defined as:

$$\mathrm{R}_i^* = \begin{cases} 0.5 + \dfrac{k - N_{undesired}}{2N_{desired}}, & if\ desired \\ \dfrac{k}{2N_{undesired}}, & if\ undesired \end{cases}$$

here the parameter *k* is the index of the solution in the Pareto rank, and rewards of undesired and desired solutions will be evenly distributed in (0, 0.5] and (0.5, 0.1], respectively.

During the generation process, for each step, *G* determines the probability of each token from the vocabulary to be chosen based on the generated sequence in previous steps. Its parameters are updated by employing a policy gradient based on the expected end reward received from the predictor. The objective function is designated as follows:

$$J(\theta) = \mathbb{E}[R^*(y_{1:T})|\theta] = \sum_{t=1}^{T} logG(y_t|y_{1:t-1}) \cdot R^*(y_{1:T})$$

By maximizing this function, the parameters θ in *G* can be optimized to ensure that *G* can construct desired SMILES sequences which can obtain the highest reward scores judged by all the *Qs*.

4.2.5. Algorithm extrapolation

Evolutionary algorithms (EAs) are common methods used in drug discovery [31]. For example, *Molecule Evoluator* is one of EAs, with mutation and crossover operations based on SMILES representation [32] for drug *de novo* design. In addition, some groups also proposed other variations of EAs [33], e.g., estimation of distribution algorithm (EDA) which is a model-based method and replaces the *mutation* and *crossover* operations with probability distribution estimation and sampling of new individuals (Fig. 4.2) [34]. Similar to EDA, *DrugEx* is a model-based method too, in which the deep learning model was employed to estimate the probability distribution of sequential decision making. However, we used a DL method to define model-based *mutation* and *crossover* operations. Moreover, we employed an RL method to replace the sample selection step for the update of model or population in EDA or EA, respectively.

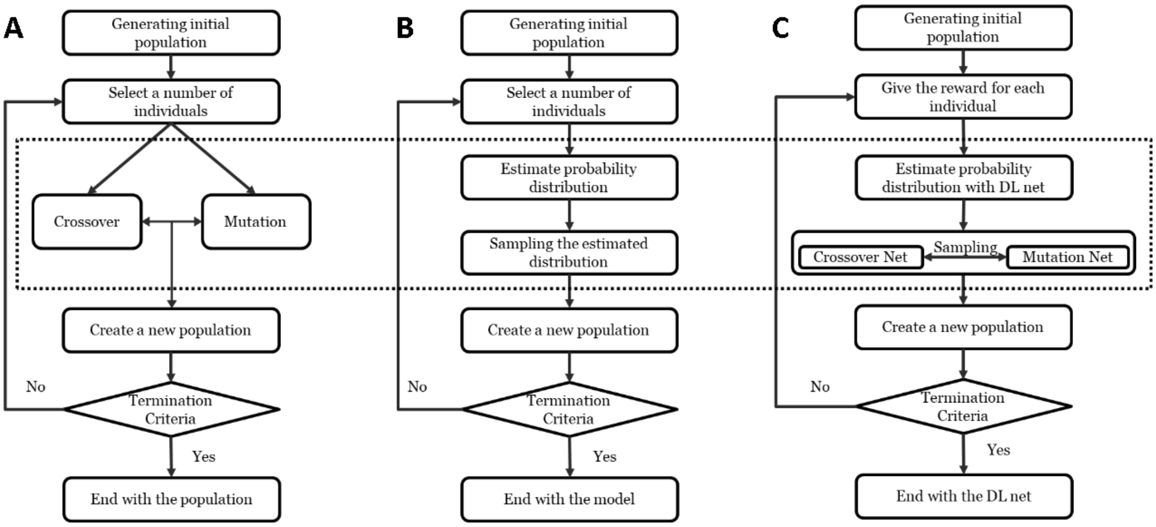

A
Generating initial population
Select a number of individuals
Crossover
Mutation
Create a new population
No
Termination Criteria
Yes
End with the population
B
Generating initial population
Select a number of individuals
Estimate probability distribution
Sampling the estimated distribution
Create a new population
No
Termination Criteria
Yes
End with the model
C
Generating initial population
Give the reward for each individual
Estimate probability distribution with DL net
Crossover Net
Sampling
Mutation Net
Create a new population
No
Termination Criteria
Yes
End with the DL net

Fig. 4.2: Flowchart comparison of evolutionary algorithm (A), estimation of distribution algorithm (B) and our proposed method (C).

4.2.6. Exploration strategy

In our previous study, we had implemented the exploration strategy through importing a fixed exploration net to enlarge the diversity of the generated molecules during the training loops. In this study, we continued to extend the methods of this exploration strategy, which resemble the *crossover* and *mutation* operations from evolutionary algorithms (EAs). Here, besides the *agent* net (G_A), we also defined exploration strategy with two other DL models: *crossover* net (G_C) and *mutation* net (G_M), which have the same RNN architecture (Fig. 4.3). The pseudo code of the exploration strategy is described in Table S4.2. Before the training process, G_M was initialized by the pre-trained model while G_A and G_C were started from the fine-tuned model. The G_M was the basic strategy employed in the previous version and its parameters were fixed and not updated during the whole training process. The G_C implemented in this work was an extended strategy whose parameters were updated iteratively based on the G_A. During the training process, each SMILES sequence was generated through combining these three RNNs: for each step, a random number from 0 to 1 is generated. If it is larger than the mutation rate (ε), the probability for token sampling is controlled by the combination of G_A and G_C, otherwise, it is determined by G_M. For each training loop, only the parameters in G_A were updated instantly based on the gradient of the RL objective function. An iteration was defined as the period of epochs after the desirability score of molecules generated by G_A did not increase. Subsequently the parameters of G_C were updated with G_A directly and the training process continued for the next iteration. The training process would continue till the percentage of desired molecules in the current iteration was not better than in the previous iterations.

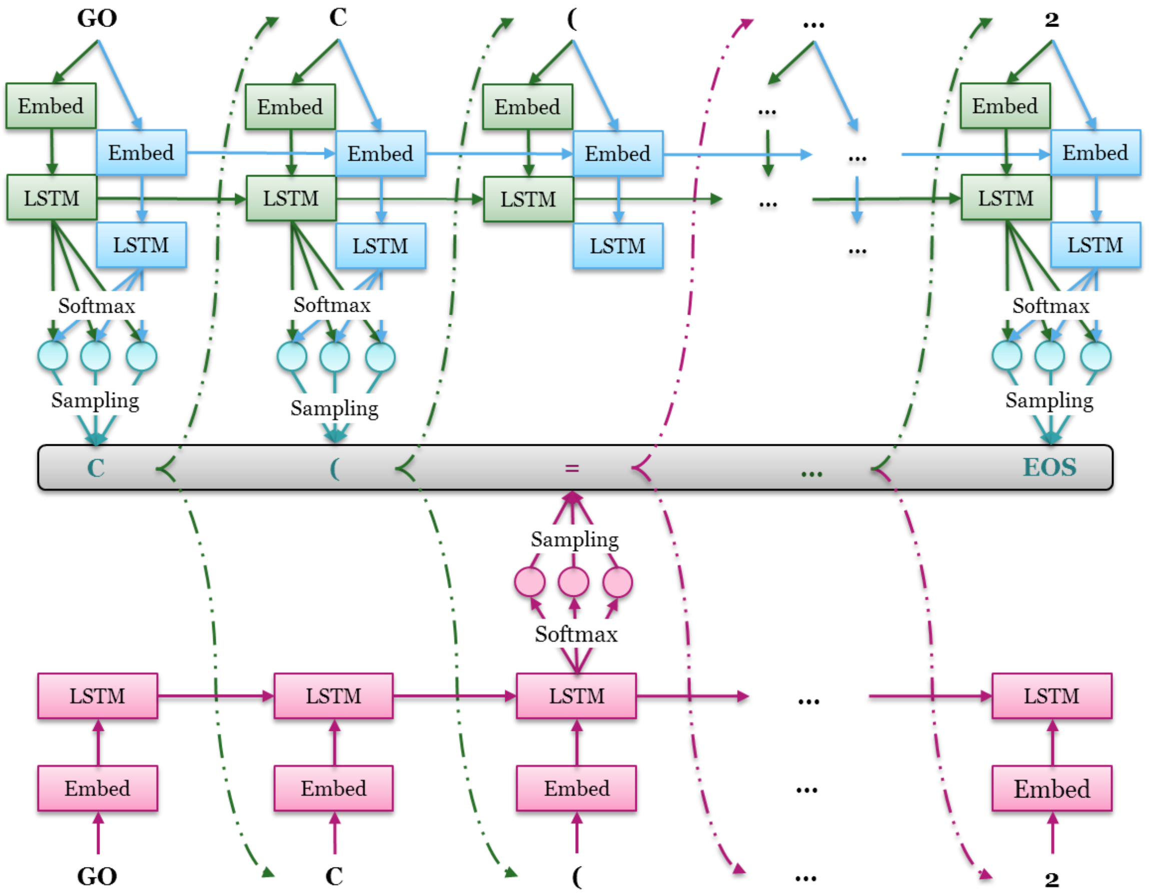


Fig. 4.3: The mechanism of updated exploration strategy, including agent net G_A, mutation net G_M (red) and crossover net G_C (blue). In the training loop, G_M is fixed, G_C is updated iteratively and G_A is trained at each epoch. For each position, a random number from 0 to 1 is generated. If it is larger than the mutation rate (ε), the probability for token sampling is controlled by the combination of G_A and G_C, otherwise, it is determined by G_M.

4.2.7. Molecular diversity

To measure molecular diversity, we adopted the metric proposed by Solow and Polasky in 1994 to estimate the diversity of a biological population in an eco-system [35]. It has been shown to be an effective method to measure the diversity of drug molecules [36]. The formula to calculate diversity was redefined to normalize the range of values from [1, m] to (0, m] as follows:

$$I(A) = \frac{1}{|A|}\boldsymbol{e}^{\intercal}F(\boldsymbol{s})^{-1}\boldsymbol{e}$$

where A is a set of drug molecules with a size of $|A|$ equal to m, $\boldsymbol{e}$ is an m-vector of 1's and

$F(\mathbf{s}) = [f(d_{ij}))]$ is a non-singular $m \times m$ distance matrix, in which $f(d_{ij})$ stands for the distance function of each pair of molecule provided as follows:

$$f(d) = e^{-\theta d_{ij}}$$

here we defined the distance d_{ij} of molecules s_i and s_j by using the Tanimoto-distance with ECFP6 fingerprints as follows:

$$d_{ij} = d(s_i, s_j) = 1 - \frac{|s_i \cap s_j|}{|s_i \cup s_j|},$$

where $| s_i \cap s_j |$ represents the number of common fingerprint bits, and $| s_i \cup s_j |$ is the number of union fingerprint bits.

4.3. Results and discussion

4.3.1. Performance of predictors

All molecules in the *LIGAND* set were used for training the QSAR models, after being transformed into predefined descriptors (2048D ECFP6 fingerprints and 19D physicochemical properties). We then tested the performance of these different algorithms with five-fold cross validation and an independent test of which the performances are shown in Fig. 4.4A-B. Here, the dataset was randomly split into five folds in the cross validation, while a temporal split with a cut-off at the year of 2015 was used for the independent test. In the cross validation test, the MT-DNN model achieved the highest value for R^2 and the lowest RMSE value for A_1AR and $A_{2A}AR$, but the RF model had the best performance for hERG based on R^2 and RMSE. However, for the independent test the RF model reached the highest R^2 and lowest RMSE across the board, although it was worse than the performance in the cross-validation test. A detailed performance overview of the RF model is shown in Fig. 4.4C-E. Because the generative model might create a large number of novel molecules, which would not be similar to the molecules in the training set, we took the robustness of the predictor into consideration. In this situation the temporal split has been shown to be more robust [19,37]. Hence the RF algorithm was chosen for constructing our environment which provides the final reward to guide the training of the generator in RL.

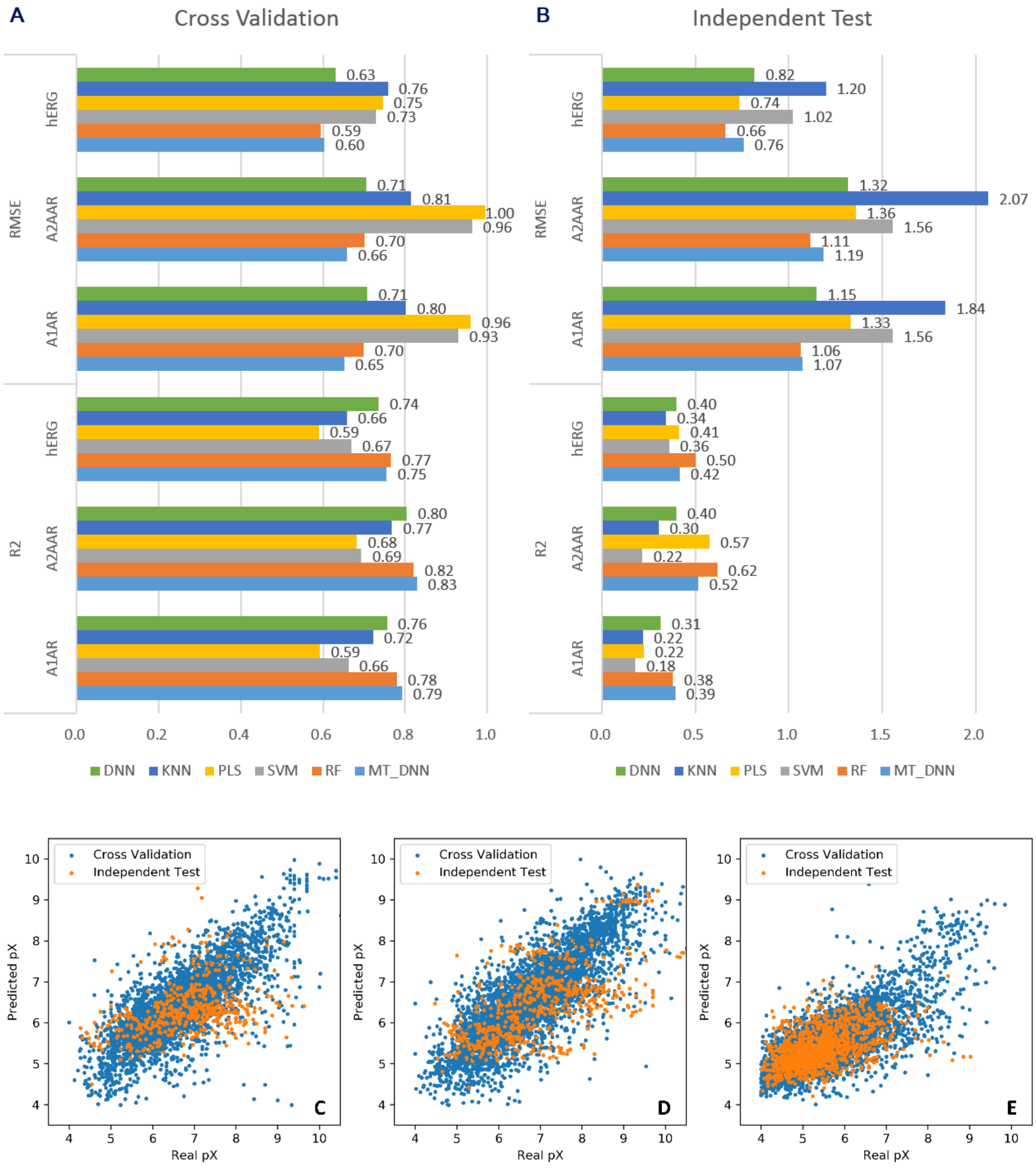


Fig. 4.4: Performance comparison of different machine learning regression models. In these two histograms (A-B), the results were obtained based on five-fold cross validation (A) and independent test (B) for the three targets. The R^2 and RMSE scores were used for evaluating the performance of different machine learning models including DNN, KNN, PLS, SVM RF and MT-DNN. In the scatter plots (C-E), each point stands for one molecule with its real pX (*x*-axis) and the predicted pX (*y*-axis) by the RF model which was chosen as the final predictors for A_1AR (C), A_{2A}AR (D) and hERG (E) based on five-fold cross validation (blue) and independent test (orange).

4.3.2. Model optimization

As in our previous work in *DrugEx v1*, we firstly pre-trained and fine-tuned the generator with the *ChEMBL* and *LIGAND* set, respectively. When testing the different types of RNNs, we analyzed the performance of the pre-trained model with 10,000 SMILES generated, and found that LSTM generated more valid SMILES (97.5%) than GRU (93.1%) which had been adopted in our previous work. Moreover, for the fine-tuning process, we split the *LIGAND* set into two subsets: training set and validation set; the validation set was not involved in parameters updating but it was essential to avoid model overfitting and to improve uniqueness of generated molecules. Subsequently 10,000 SMILES were sampled for performance evaluation. We found that the percentage valid SMILES was 97.9% for LSTM, larger than GRU with 95.7% valid SMILES, a slight improvement compared to the pre-trained model. In the end, we employed the LSTM-based pre-trained/fine-tuned models for the following investigation.

We employed the models for two cases (multi-target and target-specific) of multi-objective drug design towards three protein targets. During the training loop of *DrugEx v2*, the parameter of ε was set to different values: 10^{-2}, 10^{-3}, 10^{-4} and we also tested it without mutation net, *i.e.* the value of ε was set to 0. Generators were trained by using a policy gradient with two different rewarding schemes. After the training process converged, 10,000 SMILES were generated for each model for performance evaluation. The percentage of valid, desired, unique desired SMILES and the diversity were calculated (Table 4.2). Furthermore, we also compared the chemical space of these generated molecules with known ligands in the *LIGAND* set. Here, we employed the first two components of t-SNE on the ECFP6 descriptors of these molecules to visualize the chemical space.

4.3.3. Performance comparisons

We compared the performance of *DrugEx v2* with *DrugEx v1* and two other DL-based *de novo* drug design methods: *REINVENT* [38] and *ORGANIC* [39]. In order to make a fair benchmark, we trained these four methods with the same environments to provide the unified predicted bioactivity scores for each of the generated molecules. It should be mentioned that these methods are all SMILES-based RNNs generators but trained under different RL frameworks. Therefore, these generators were constructed with the same RNN structures of and initialized with the same pre-trained/fine-tuned models. We also tested *REINVENT* 2.0 [40] but found the training loop did not converge in the PF scheme. We speculate this is due to the number of desired molecules generated by the initial state of the model being too small, not containing enough information. Moreover, addition of a scaffold filter is repetitive when integrated into thePF scheme because it is similar to the similarity-based crowding distance algorithm in the PF scheme. Finally, a scaffold filter is a hard condition, because it directly penalizes the score of similar molecules to 0 while the PF scheme decreased the similar molecules. Hence we have not shown these results here.

In the WS scheme we did not choose fixed weights for objectives but dynamic values which can be adjusted automatically during the training process. The reason for this is that if the fixed weights should be optimized as the hyperparameters, which would be more time consuming. Moreover, the distribution of scores for each objective was not comparable. If the affinity score was required to be higher, few of the molecules generated by the model with the initial state were satisfactory, but if a lower affinity score was required, most of the generated molecules by the pre-trained/fine-tuned model met this need without further training of RL. Therefore, weights were set as dynamic parameters and determined by the ratio between desired and undesired molecules generated by the model at the current training step. This approach ensures that the objectives with lower scores would get more importance than others during the training loop to balance the different objectives and generate more desired molecules.

The performance of the model with different ε is shown in Table S4.3. A higher ε generates molecules with larger diversity but low desirability compared to a lower ε in both multi-target and target-specific cases. In addition, an appropriate ε guarantees the model generates molecules which have a more similar distribution of important substructures with the desired ligands in the *LIGAND* set (Fig. S4.1). With the WS scheme, the model generates molecules with a high desirability, but the diversity is lower than the desired ligands in the training set. On the contrary, the PF scheme helped the model generate molecules with a larger diversity than the ligands in the training set, but the desirability was not as high as in the WS rewarding scheme. Moreover, the generated molecules in the PF scheme have more similar distribution of substructures to the *LIGAND* set than in the WS scheme.

Table 4.2: Comparison of the performance of the different methods in the multi-target case.

Rewarding Scheme	**Dataset**	**Validity**	**Desirability**	**Uniqueness**	**Diversity**	**Purine Ring**	**Furan Ring**	**Benzene Ring**
	LIGAND	100.00%	12.40%	100.00%	0.66	21.30%	35.44%	79.24%
PF	*DrugEx v1*	98.28%	43.27%	88.96%	0.71	**17.37%**	41.05%	80.95%
	DrugEx v2	99.57%	**80.81%**	87.29%	0.7	13.97%	**32.01%**	**80.26%**
	ORGANIC	**98.84%**	66.01%	82.67%	0.65	17.27%	56.38%	68.87%
	REINVENT	99.54%	57.43%	**98.84%**	**0.77**	0.64%	40.38%	92.05%
WS	*DrugEx v1*	97.76%	38.44%	93.44%	0.71	**10.76%**	**36.42%**	86.99%
	DrugEx v2	**99.80%**	**97.45%**	89.08%	0.49	3.63%	21.06%	96.18%
	ORGANIC	99.08%	61.10%	77.65%	0.68	9.08%	70.99%	**83.91%**
	REINVENT	99.54%	70.98%	**99.11%**	**0.71**	0.04%	23.23%	96.28%

Shown are validity, desirability, uniqueness, and substructure distributions of SMILES generated by four different methods in the multi-target case with PF and WS rewarding schemes. For the validity, desirability and uniqueness, the highest values are bold, while for the distribution of substructures, the bold data are labeled as the most closed to the values in the *LIGAND* set.

In the multi-target case, these four methods with different rewarding schemes show similar performance, *i.e.* the WS scheme can help models improve the desirability while the PF scheme assists models to achieve better diversity and distribution of substructures (Table 4.2). Here, *REINVENT* with the PF scheme achieved the largest diversity, whereas *DrugEx v1* had the most similar substructure distribution to the molecules in the *LIGAND* set, and *DrugEx v2* achieved the best desirability with both PR and WS schemes compared to the three other algorithms. The diversity and distribution of substructures were also most similar to the best results. In addition, in the target-specific case results were similar to the multi-target case, (Table 4.3), and for the distribution of purine and furan rings, *DrugEx v2* surpassed v1 to be most similar to the *LIGAND* set. When investigating the SA and QED scores, we observed that the PF scheme helped all of generated molecules being more drug-like because of higher QED scores than the WS scheme in both multi-target case (Fig. 4.5A-D) and target-specific case (Fig. 4.5E-H). In comparison of these methods, the molecules generated by *REINVENT* were supposedly easier to be synthesized and more drug-like than others, but the molecules of *DrugEx v1* had more similar distributions with the molecules in the *LIGAND* set.

Table 4.3: Comparison of the performance of the different methods in the target-specific case.

Rewarding Scheme	**Dataset**	**Validity**	**Desirability**	**Uniqueness**	**Diversity**	**Purine Ring**	**Furan Ring**	**Benzene Ring**
	LIGAND	100.00%	14.63%	100.00%	0.67	28.27%	50.61%	71.84%
PF	*DrugEx v1*	98.07%	48.42%	87.32%	0.73	**29.65%**	61.61%	**70.99%**
	DrugEx v2	99.53%	**89.49%**	90.55%	0.73	23.73%	**56.23%**	67.40%
	ORGANIC	98.29%	86.98%	80.30%	0.64	10.60%	89.27%	65.28%
	REINVENT	**99.59%**	70.66%	**99.33%**	**0.79**	3.85%	33.82%	92.53%
WS	*DrugEx v1*	97.61%	44.96%	95.89%	**0.68**	78.92%	**80.21%**	68.02%
	DrugEx v2	**99.62%**	**97.86%**	90.54%	0.31	19.58%	98.56%	51.87%
	ORGANIC	98.97%	88.14%	84.13%	0.49	9.68%%	96.66%	**71.48%**
	REINVENT	99.55%	81.27%	98.87%	0.34	**25.13%**	97.52%	74.61%

Shown are validity, desirability, uniqueness, and substructure distributions of SMILES generated by four different methods in the target-specific case with PF and WS rewarding schemes. For the validity, desirability and uniqueness, the highest values are bold, while for the distribution of substructures, the bold data are labeled as the most closed to the values in the *LIGAND* set.

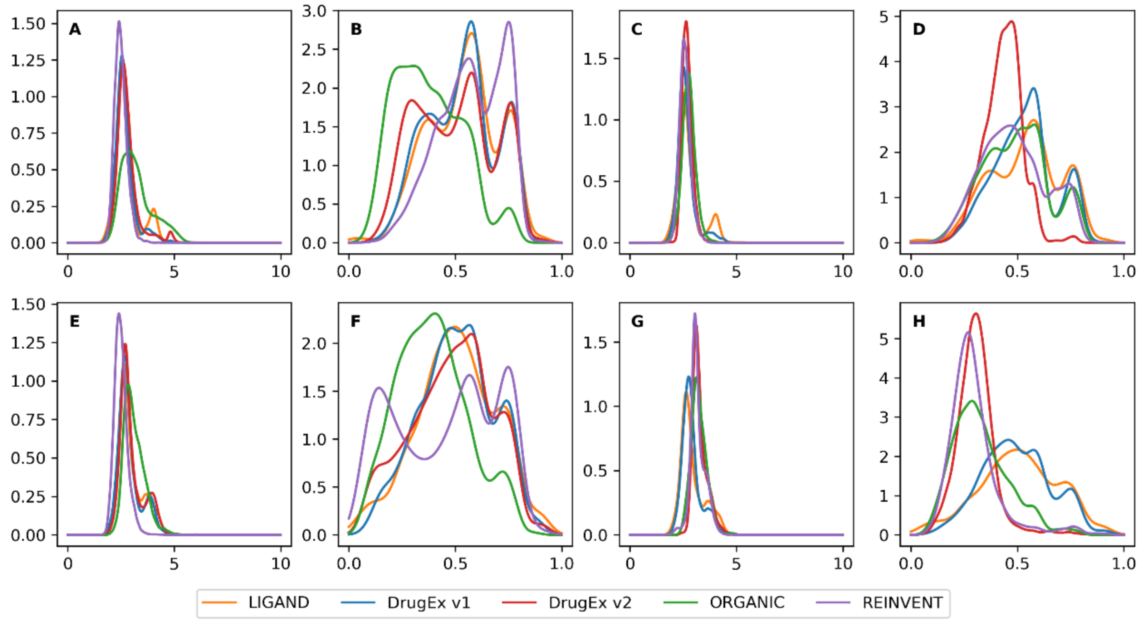


Fig. 4.5: the distribution of SA score and QED score of desired ligands in the *LIGAND* set and of molecules generated by four different methods with PR (A, B, E and F) and WS (C, D, G and H) rewarding schemes in the multi-target case (A-D) and target-specific case (E-H). The molecules from the *LIGAND* set were shown as color of orange, and the molecules generated by *DrugEx v1*, *v2*, *ORGANIC* and *REINVENT* were represented with colors of blue, green, red, and purple, respectively. Overall DrugEx v1 and v2 are better able to emulate the observed distributions in the training set compared to *ORGANIC* and *REINVENT*.

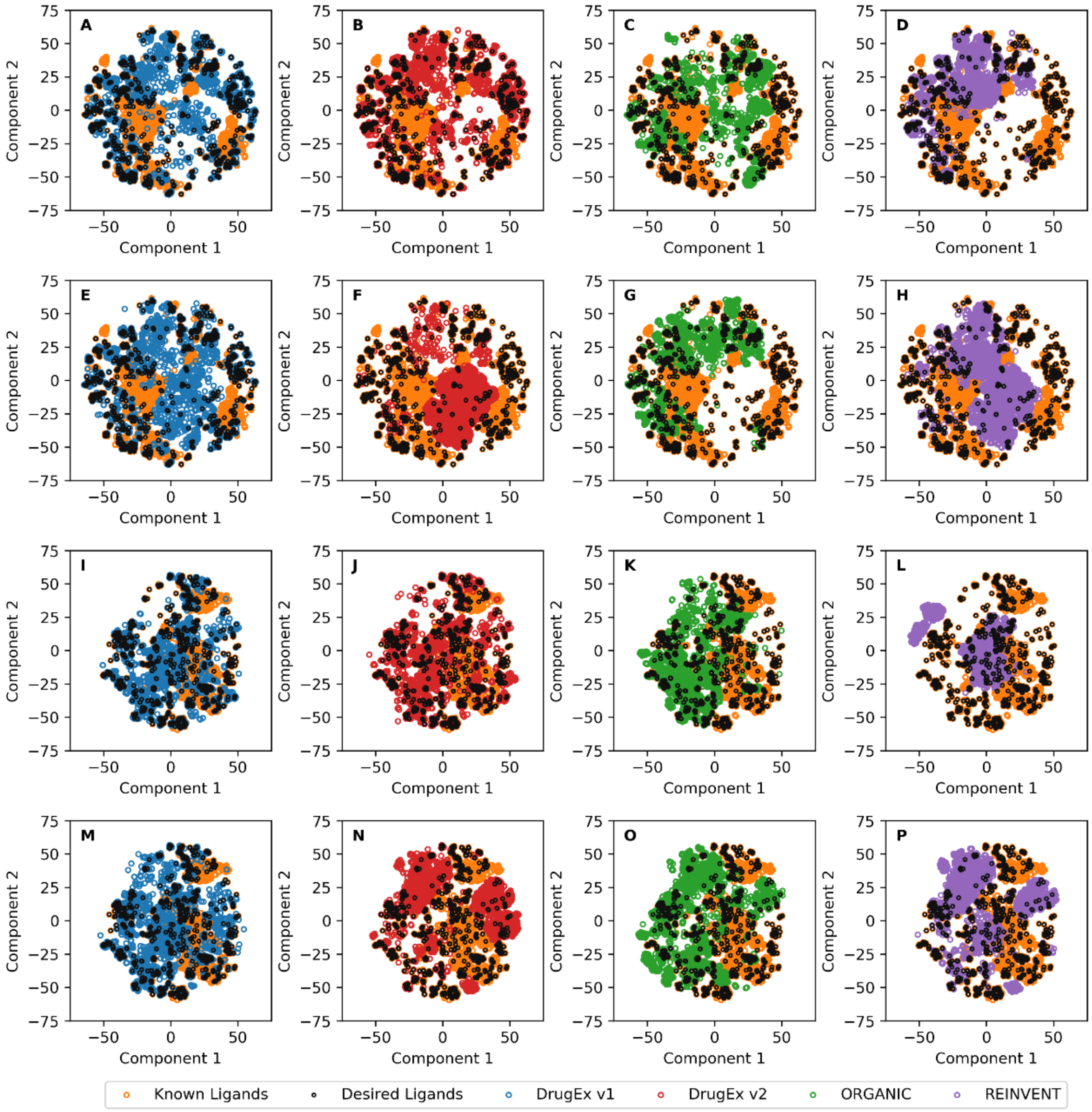


Fig. 4.6: Comparison of the chemical space of the *LIGAND* set and generated molecules. Shown are all known ligands (orange) and desired molecules (black). Moreover shown are generated molecules by *DrugEx v1* (A, E, I, M, blue), *v2* (B, F, J, N, red), *ORGANIC* (C, G, K, O, green) and *REINVENT* (D, H, L, P, purple). Distinction can be made between the multi-target case (A-H) and target specific case (I-P). Additionally the distinction can be made between PF scheme based scoring (A-D and I-L) and WS scheme based scoring (E-H and M-P). Chemical space is represented by the first two components in t-SNE with ECFP6 descriptors of molecules. Similar to our previous work it can be seen that *DrugEx* better covers the whole chemical space of the input data. In particular in the multi-target case with a Pareto optimization based scoring function (E-H) the improved coverage in all sections, including isolated active ligands, becomes clear.

With respect to chemical space, we employed t-SNE with the ECFP6 descriptors of all molecules for both multi-target (Fig. 4.6A-H) and target-specific cases (Fig. 4.6I-P). In the multi-target case, most of the desired ligands in the *LIGAND* set were distributed in the margin and PF scheme could guide all of the generators to better cover chemical space than WS scheme. In the target-specific case, the desired ligands in the *LIGAND* set were distributed more dispersed in both of the margin and the center regions. For both of these two cases, only part of the region occupied by desired ligands in the *LIGAND* set were overlapped with *REINVENT* and *ORGANIC*, but almost all of it is covered by *DrugEx v1* and *v2*. Especially, in contrast to WS scheme *DrugEx v2* had a significant improvement of chemical space coverage with PF scheme. Hence in this case, the PF scheme could not guide all generators better in the target-specific case regarding coverage compared to WS scheme except for *DrugEx v2*. A possible reason is that the molecules generated by *DrugEx v1* and *v2* offer a more similar distribution of substructures to desired ligands in the *LIGAND* set than *REINVENT* and *ORGANIC*.

As an example, 16 possible antagonists (without ribose moiety and molecular weight < 500) generated by *DrugEx v2* with PF scheme were selected as candidates for both multi-target cases and target specific case, respectively. These molecules were ordered by the selectivity which was calculated as the difference of pXs between two different protein targets. In the multi-target cases (Fig. 4.7A), because the desired ligands prefer A_1AR and A_{2A}AR to hERG, the row and column is the selectivity of A_{2A}AR and A_1AR against hERG, respectively, while the generated molecules are required to bind only A_{2A}AR rather than A_1AR and hERG in the target-specific case (Fig. 4.7B), selectivity of A_{2A}AR against A_1AR and hERG were represented as the row and column, respectively.

In order to prove the effectiveness of our proposed method, we tested it with 20 goal-directed molecule generation tasks on the GuacaMol benchmark platform [41]. These tasks contain different requirements, including similarity, physicochemical properties, isomerism, scaffold matching, *etc.* The detailed description of these tasks is provided in ref [41] and our results are shown in Table S4.4. We pre-trained our model with the dataset provided by the GuacaMol platform, in which all molecules from the ChEMBL database are included and similar molecules to the target ligands in the tasks were removed. Then we choose the top 1024 molecules in the training set to fine-tune our model for each task, before reinforcement learning was started. Our method scores the best in 12 out of 20 tasks compared with the baseline models provided by the GuacaMol platform, leading to an overall second place. Moreover, the performance between the LSTM benchmark method and our methods were similar in these tasks, possibly because they have similar architectures of neural networks. All in all, this benchmark demonstrated that our proposed method has improved generality for drug *de novo* design tasks. It is worth being mentioned that our method is not effective enough yet for some tasks with contradictory objectives in the narrow chemical space. The main reason is that our method emphasizes to obtain a large number of feasible molecules to occupy the diverse chemical space rather than a small number of optimal molecules to achieve the highest score. For example, in the *Sitagliptin MPO task*, the aim is finding molecules which are dissimilar to sitagliptin but have a similar molecular formula to sitagliptin, and our method was not as good as Graph GA, which is a graph-based genetic algorithm.

4.4. Conclusion and future prospect

In this work, we proposed a Pareto-based multi-objective learning algorithm for drug *de novo* design towards multiple targets based on different requirements of affinity scores for multiple targets. We transferred the concept of an evolutionary algorithm (including mutation and crossover operations) into RL to update *DrugEx* for multi-objective optimization. In addition, Pareto ranking algorithms were also integrated into our model to handle the contradictory objectives common in drug discovery and enlarge the chemical diversity. In order to prove effectiveness, we tested the performance of *DrugEx v2* in both multi-target and target-specific cases. We found that a large percentage of generated SMILES were valid and desired molecules without many duplications. Moreover, generated molecules were also similar to known ligands and covered almost every corner of the chemical space that known ligands occupy, which could not be repeated by tested competing methods. In addition to our work here other methods to improve the diversity of generated molecules were proposed such as REINVENT 2.0 [40]. In addition, some other teams also trained the new deep learning model (e.g. BERT, Transformer, GPT2) with a larger dataset and achieved better results [42,43]. In future work, we will continue to update *DrugEx* with these new deep learning models to deal with different molecular representations, such as graphs or fragments [31]. We will also integrate more objectives (e.g. stability, synthesizability), especially when these objectives are contradictory, such that the model allows user-defined weights for each objective to generate more reliable candidate ligands and better steer the generative process.

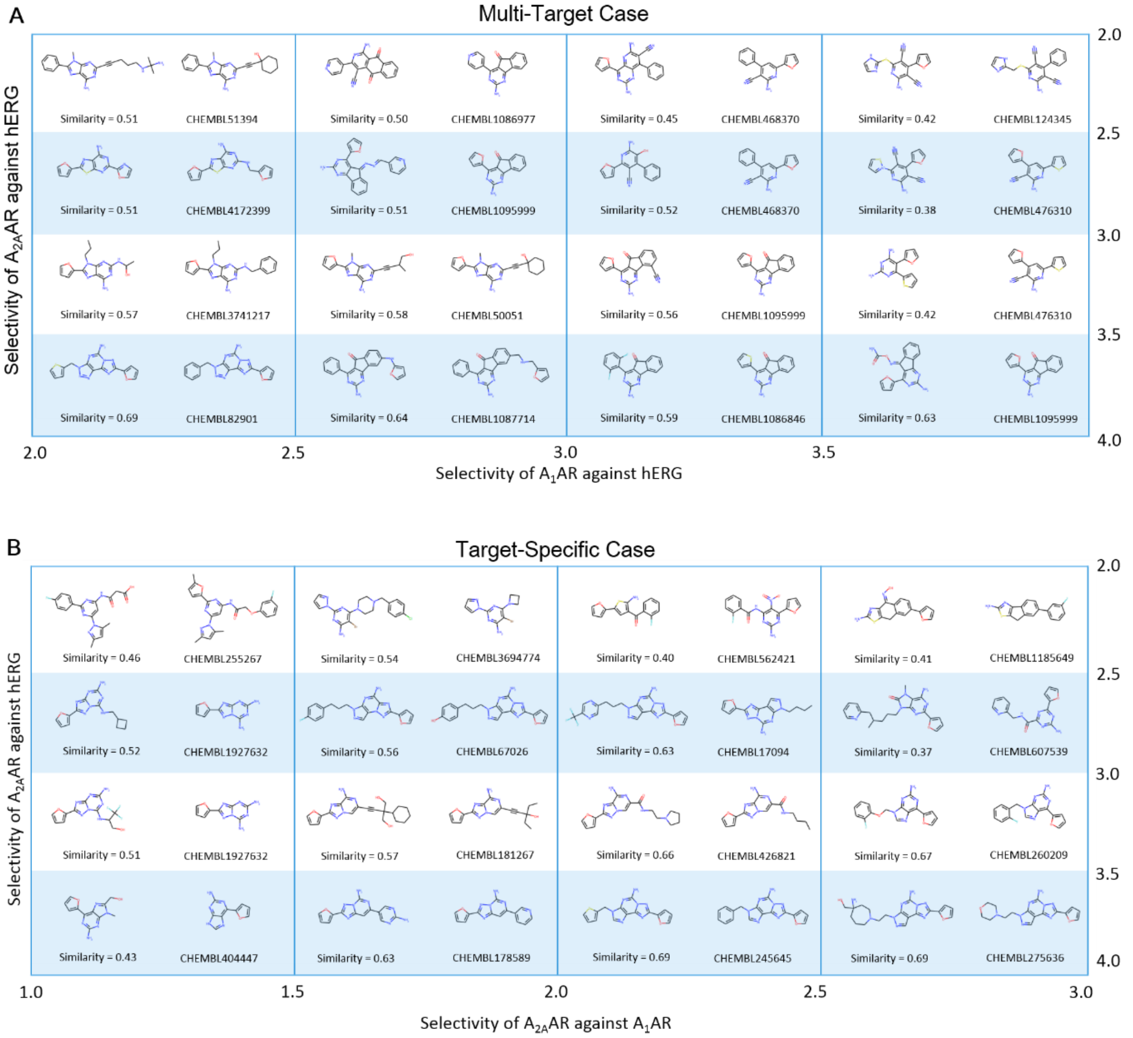


Fig. 4.7: Some candidate molecules were selected from molecules generated by *DrugEx v2* with the PF scheme for both multi-target case and target-specific case. In multi-target case (A), these molecules were ordered by the selectivity of A_1AR and A_{2A}AR against hERG as *x*-axis and *y*-axis, respectively. In target-specific case (B), these molecules were ordered by the selectivity of A_{2A}AR against A_1AR and hERG as *x* and *y*-axis, respectively. For each cell, the structure at the left is the generated molecule labeled with its similarity to the most similar ligands in the *LIGAND* set, located at the right and labeled with their ChEMBL ID.

Declarations

Availability of data and materials

The data used in this study is publicly available ChEMBL data, the algorithm published in this manuscript is made available at https://github.com/XuhanLiu/DrugEx.

Authors' Contributions

XL and GJPvW conceived the study and performed the experimental work and analysis. KY, APIJ, ME and HWTvV provided feedback and critical input. All authors read, commented on and approved the final manuscript.

Acknowledgements

XL thanks Chinese Scholarship Council (CSC) for funding, GJPvW thanks the Dutch Research Council and Stichting Technologie Wetenschappen (STW) for financial support (STW-Veni #14410).

Competing Interests

The authors declare that they have no competing interests

References

1. Chaudhari R, Tan Z, Huang B, Zhang S (2017) Computational polypharmacology: a new paradigm for drug discovery. Expert Opin Drug Discov 12 (3):279-291. doi:10.1080/17460441.2017.1280024
2. Giacomini KM, Krauss RM, Roden DM, Eichelbaum M, Hayden MR, Nakamura Y (2007) When good drugs go bad. Nature 446 (7139):975-977. doi:10.1038/446975a
3. Lounkine E, Keiser MJ, Whitebread S, Mikhailov D, Hamon J, Jenkins JL, Lavan P, Weber E, Doak AK, Cote S, Shoichet BK, Urban L (2012) Large-scale prediction and testing of drug activity on side-effect targets. Nature 486 (7403):361-367. doi:10.1038/nature11159
4. Cook D, Brown D, Alexander R, March R, Morgan P, Satterthwaite G, Pangalos MN (2014) Lessons learned from the fate of AstraZeneca's drug pipeline: a five-dimensional framework. Nat Rev Drug Discov 13 (6):419-431. doi:10.1038/nrd4309
5. Siramshetty VB, Nickel J, Omieczynski C, Gohlke BO, Drwal MN, Preissner R (2016) WITHDRAWN--a resource for withdrawn and discontinued drugs. Nucleic Acids Res 44 (D1):D1080-1086. doi:10.1093/nar/gkv1192
6. Hopkins AL (2008) Network pharmacology: the next paradigm in drug discovery. Nat Chem Biol 4 (11):682-690. doi:10.1038/nchembio.118
7. Anighoro A, Bajorath J, Rastelli G (2014) Polypharmacology: challenges and opportunities in drug discovery. J Med Chem 57 (19):7874-7887. doi:10.1021/jm5006463
8. van Westen GJ, Wegner JK, Geluykens P, Kwanten L, Vereycken I, Peeters A, Ijzerman AP, van Vlijmen HW, Bender A (2011) Which compound to select in lead optimization? Prospectively validated proteochemometric models guide preclinical development. PLoS One 6 (11):e27518. doi:10.1371/journal.pone.0027518
9. Csermely P, Agoston V, Pongor S (2005) The efficiency of multi-target drugs: the network approach might help drug design. Trends Pharmacol Sci 26 (4):178-182. doi:10.1016/j.tips.2005.02.007
10. Fredholm BB (2010) Adenosine receptors as drug targets. Exp Cell Res 316 (8):1284-1288. doi:10.1016/j.yexcr.2010.02.004
11. Fredholm BB, IJzerman AP, Jacobson KA, Linden J, Muller CE (2011) International Union of Basic and Clinical Pharmacology. LXXXI. Nomenclature and classification of adenosine receptors--an update. Pharmacol Rev 63 (1):1-34. doi:10.1124/pr.110.003285
12. Chen JF, Eltzschig HK, Fredholm BB (2013) Adenosine receptors as drug targets--what are the challenges? Nat Rev Drug Discov 12 (4):265-286. doi:10.1038/nrd3955
13. Trudeau MC, Warmke JW, Ganetzky B, Robertson GA (1995) HERG, a human inward rectifier in the voltage-gated potassium channel family. Science 269 (5220):92-95. doi:10.1126/science.7604285
14. Milnes JT, Crociani O, Arcangeli A, Hancox JC, Witchel HJ (2003) Blockade of HERG potassium currents by fluvoxamine: incomplete attenuation by S6 mutations at F656 or Y652. Br J Pharmacol 139 (5):887-898. doi:10.1038/sj.bjp.0705335
15. Sanguinetti MC, Tristani-Firouzi M (2006) hERG potassium channels and cardiac arrhythmia. Nature 440 (7083):463-469. doi:10.1038/nature04710
16. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521 (7553):436-444. doi:10.1038/nature14539
17. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T (2018) The rise of deep learning in drug discovery. Drug discovery today. doi:10.1016/j.drudis.2018.01.039
18. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, Light Y, McGlinchey S, Michalovich D, Al-Lazikani B, Overington JP (2012) ChEMBL: a large-scale bioactivity database

for drug discovery. Nucleic Acids Res 40 (Database issue):D1100-1107. doi:10.1093/nar/gkr777

19. Lenselink EB, Ten Dijke N, Bongers B, Papadatos G, van Vlijmen HWT, Kowalczyk W, IJzerman AP, van Westen GJP (2017) Beyond the hype: deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set. Journal of cheminformatics 9 (1):45. doi:10.1186/s13321-017-0232-0
20. Liu X, Ye K, van Vlijmen HWT, IJzerman AP, van Westen GJP (2019) An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine A2A receptor. Journal of cheminformatics 11 (1):35. doi:10.1186/s13321-019-0355-6
21. RDKit: Open-Source Cheminformatics Software. http://www.rdkit.org.
22. Rogers D, Hahn M (2010) Extended-connectivity fingerprints. Journal of chemical information and modeling 50 (5):742-754. doi:10.1021/ci100050t
23. Scikit-Learn: machine learning in Python. http://www.scikit-learn.org/.
24. PyTorch. https://pytorch.org/.
25. Kingma DP, Ba J (2014) Adam: A Method for Stochastic Optimization. arXiv:1412.6980
26. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv:1412.3555
27. Ertl P, Schuffenhauer A (2009) Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. Journal of cheminformatics 1 (1):8. doi:10.1186/1758-2946-1-8
28. Bickerton GR, Paolini GV, Besnard J, Muresan S, Hopkins AL (2012) Quantifying the chemical beauty of drugs. Nat Chem 4 (2):90-98. doi:10.1038/nchem.1243
29. Deb K, Agrawal S, Pratap A, Meyarivan T A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II. In: Schoenauer M, Deb K, Rudolph G et al. (eds) Parallel Problem Solving from Nature PPSN VI, Berlin, Heidelberg, 2000// 2000. Springer Berlin Heidelberg, pp 849-858
30. Emmerich MTM, Deutz AH (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Nat Comput 17 (3):585-609. doi:10.1007/s11047-018-9685-y
31. Liu X, IJzerman AP, van Westen GJP (2021) Computational Approaches for De Novo Drug Design: Past, Present, and Future. Methods Mol Biol 2190:139-165. doi:10.1007/978-1-0716-0826-5_6
32. Lameijer EW, Kok JN, Back T, IJzerman AP (2006) The molecule evoluator. An interactive evolutionary algorithm for the design of drug-like molecules. Journal of chemical information and modeling 46 (2):545-552. doi:10.1021/ci050369d
33. van der Horst E, Marques-Gallego P, Mulder-Krieger T, van Veldhoven J, Kruisselbrink J, Aleman A, Emmerich MT, Brussee J, Bender A, IJzerman AP (2012) Multi-objective evolutionary design of adenosine receptor ligands. Journal of chemical information and modeling 52 (7):1713-1721. doi:10.1021/ci2005115
34. Nicolaou CA, Brown N (2013) Multi-objective optimization methods in drug design. Drug Discov Today Technol 10 (3):e427-435. doi:10.1016/j.ddtec.2013.02.001
35. Solow AR, Polasky S (1994) Measuring biological diversity. Environmental and Ecological Statistics 1 (2):95-103. doi:10.1007/BF02426650
36. Yevseyeva I, Lenselink EB, de Vries A, IJzerman AP, Deutz AH, Emmerich MTM (2019) Application of portfolio optimization to drug discovery. Information Sciences 475:29-43. doi: 10.1016/j.ins.2018.09.049
37. Sheridan RP (2013) Time-split cross-validation as a method for estimating the goodness of prospective prediction. Journal of chemical information and modeling 53 (4):783-790. doi:10.1021/ci400084k

38. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. Journal of cheminformatics 9 (1):48. doi:10.1186/s13321-017-0235-x

39. Benjamin S-L, Carlos O, Gabriel L. G, Alan A-G (2017) Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). doi:10.26434/chemrxiv.5309668.v3

40. Blaschke T, Arus-Pous J, Chen H, Margreitter C, Tyrchan C, Engkvist O, Papadopoulos K, Patronov A (2020) REINVENT 2.0: An AI Tool for De Novo Drug Design. Journal of chemical information and modeling 60 (12):5918-5922. doi:10.1021/acs.jcim.0c00915

41. Brown N, Fiscato M, Segler MHS, Vaucher AC (2019) GuacaMol: Benchmarking Models for de Novo Molecular Design. Journal of chemical information and modeling 59 (3):1096-1108. doi:10.1021/acs.jcim.8b00839

42. Wang S, Guo Y, Wang Y, Sun H, Huang J (2019) SMILES-BERT: Large Scale Unsupervised Pre-Training for Molecular Property Prediction. Paper presented at the Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Niagara Falls, NY, USA, , 429–436. doi: 10.1145/3307339.3342186

43. Chithrananda S, Grand G, Ramsundar BJae-p (2020) ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction.arXiv:2010.09885

Table S4.1: All tokens in vocabulary for SMILES sequence construction with RNN model.

Atoms							**Bonds**	**Controls**		
Common Atoms					**Aromatic Atoms**		**--**	**Rings**	**Branchs**	**On-Off**
B	[As+]	[CH-]	[N]	[SH2]	[b-]	[se+]	-	1	(	GO
C	[As]	[CH2]	[O+]	[SH]	[c+]	[se]	=	2	)	EOS
F	[B-]	[CH]	[O-]	[Se+]	[c-]	[te+]	#	3		
I	[BH-]	[I+]	[OH+]	[SeH]	[cH-]	[te]		4		
L	[BH2-]	[IH2]	[O]	[Se]	[n+]	b		5		
N	[BH3-]	[N+]	[P+]	[SiH2]	[n-]	c		6		
O	[B]	[N-]	[PH]	[SiH]	[nH+]	n		7		
P	[C+]	[NH+]	[S+]	[Si]	[nH]	o		8		
R	[C-]	[NH-]	[S-]	[Te]	[o+]	p		9		
S		[NH2+]	[SH+]		[s+]	s				

Considering that the sterochemical information of molecules and ionic bonds were ignored, we removed the “@”, “\”, “/”, “.”.

Table S4.2: The pseudo code of exploration strategy in *DrugEx v2*

```
Algorithm explore:
    Input:
        $G_A$: Agent net, $G_C$: Crossover net, $G_M$: Mutation net,
        ε: mutation rate, size: number of generated molecules
        vocab: vocabulary of tokens which is consisted of SMILES sequence.
    Output:
        samples: a list of generated SMILES sequences

    samples ← []
    For i ← 1 to size:
        sample ← []
        token ← ‘GO’
        h ← INIT_STATES ()
        mutate ← RANDOM_FLOAT (0, 1)
        ratio ← RANDOM_FLOAT (0, 1)
        For step ← 1 to max_lenth:
            $prob_A$, $h_A$ ← $G_A$ (t, $h_A$)
            $prob_C$, $h_C$ ← $G_C$ (t, $h_C$)
            $prob_M$, $h_C$ ← $G_M$ (t, $h_M$)
            If ε > mutate Then
                prob ← $prob_M$
            Else
                prob ← $prob_A$ * ratio + $prob_M$ * ratio
            token ← DISTRIBUTION_BASED_SAMPLING (prob, vocab)
            insert token to sample
            If token == ‘EOS’ Then
                Insert sample to samples
                Break
        End
    End
    Return samples
```

Table S4.3: Comparison of validity, desirability, uniqueness and substructures distributions of SMILES generated by *DrugEx v2* with different *ε* in the multi-target and target-specific cases by using PF and WS rewarding schemes, respectively.

Case	Reward Scheme	Dataset / ε	Validity	Desirability	Uniqueness	Diversity	Purine Ring	Furan Ring	Benzene Ring
Multi-Target Case		*LIGAND*	100.00%	14.63%	100.00%	0.67	21.30%	35.44%	79.24%
	PF	10^{-2}	99.39%	71.37%	**90.47%**	**0.72**	12.39%	34.69%	82.05%
		10^{-3}	99.57%	80.81%	88.96%	0.71	**13.97%**	32.01%	**80.26%**
		10^{-4}	**99.72%**	**83.86%**	87.19%	0.71	12.45%	30.58%	84.04%
		0	99.47%	73.76%	84.41%	0.70	13.35%	**35.71%**	81.89%
	WS	10^{-2}	99.54%	87.56%	93.08%	**0.60**	**9.66%**	**28.83%**	92.19%
		10^{-3}	**99.80%**	97.45%	93.44%	0.49	3.63%	21.06%	96.18%
		10^{-4}	99.79%	**98.15%**	**93.56%**	0.53	2.89%	24.95%	**91.46%**
		0	99.78%	98.00%	90.19%	0.49	5.02%	16.45%	96.77%
Target-Specific Case		*LIGAND*	100.00%	12.40%	100.00%	0.66	28.27%	50.61%	71.84%
	PF	10^{-2}	99.48%	88.76%	**91.98%**	**0.77**	18.31%	**47.50%**	68.95%
		10^{-3}	99.53%	89.49%	87.32%	0.72	23.73%	56.23%	67.40%
		10^{-4}	**99.55%**	**91.84%**	88.31%	0.74	**26.86%**	39.68%	74.36%
		0	99.54%	91.47%	88.94%	0.75	22.95%	43.08%	**71.50%**
	WS	10^{-2}	99.16%	86.45%	93.97%	**0.42**	**42.84%**	97.26%	**72.45%**
		10^{-3}	99.62%	**97.86%**	**95.89%**	0.31	60.81%	98.56%	51.87%
		10^{-4}	**99.67%**	96.82%	94.56%	0.34	55.14%	**93.69%**	45.40%
		0	99.33%	96.28%	92.60%	0.35	42.86%	98.34%	63.47%

For the validity, desirability and uniqueness, the largest data is bold, while for the distribution of substructures, the bold data are labeled as the most closed to the values in the *LIGAND* set.

Table S4.4: Results of the Goal-Directed tasks for our proposed method *DrugEx v2* and other baseline models on GuacaMol Benchmark.

Benchmark	Best of Dataset	SMILES GA	Graph MCTS	Graph GA	SMILES LSTM	DrugEx v2
Celecoxib rediscovery	0.505	0.732	0.355	**1**	**1**	**1**
Troglitazone rediscovery	0.419	0.515	0.311	**1**	**1**	**1**
Thiothixene rediscovery	0.456	0.598	0.311	**1**	**1**	**1**
Aripiprazole similarity	0.595	0.834	0.38	**1**	**1**	**1**
Albuterol similarity	0.719	0.907	0.749	**1**	**1**	**1**
Mestranol similarity	0.629	0.79	0.402	**1**	**1**	**1**
C11H24	0.684	0.829	0.41	0.971	**0.993**	**0.993**
C9H10N2O2PF2Cl	0.747	0.889	0.631	0.982	0.879	**1**
Median molecules 1	0.334	0.334	0.225	0.406	**0.438**	0.418
Median molecules 2	0.351	0.38	0.17	0.432	0.422	**0.435**
Osimertinib MPO	0.839	0.886	0.784	0.953	0.907	**0.967**
Fexofenadine MPO	0.817	0.931	0.695	**0.998**	0.959	0.942
Ranolazine MPO	0.792	0.881	0.616	**0.92**	0.855	0.909
Perindopril MPO	0.575	0.661	0.385	0.792	0.808	**0.812**
Amlodipine MPO	0.696	0.722	0.533	0.894	0.894	**0.898**
Sitagliptin MPO	0.509	0.689	0.458	**0.891**	0.545	0.517
Zaleplon MPO	0.547	0.413	0.488	**0.754**	0.669	0.693
Valsartan SMARTS	0.259	0.552	0.04	**0.99**	0.978	0.978
Scaffold Hop	0.933	0.97	0.59	**1**	0.996	0.989
Deco Hop	0.738	0.885	0.478	**1**	0.998	0.986
Total	12.144	14.398	9.011	**17.983**	17.341	17.537

GucacaMol platform contains 20 tasks with different requirements, including smilarity, physicochemical properties, isomerism, scaffold matching, *etc.*. The results for baseline models were cited from ref [41]. The bold data are shown as the best result for each task achieved by different methods.

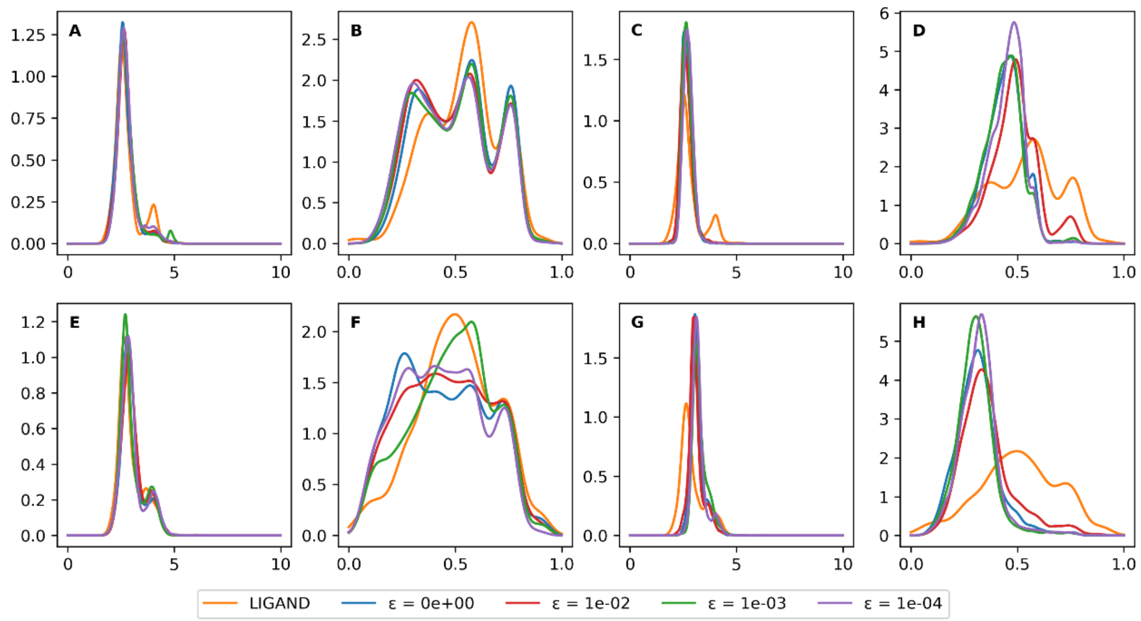
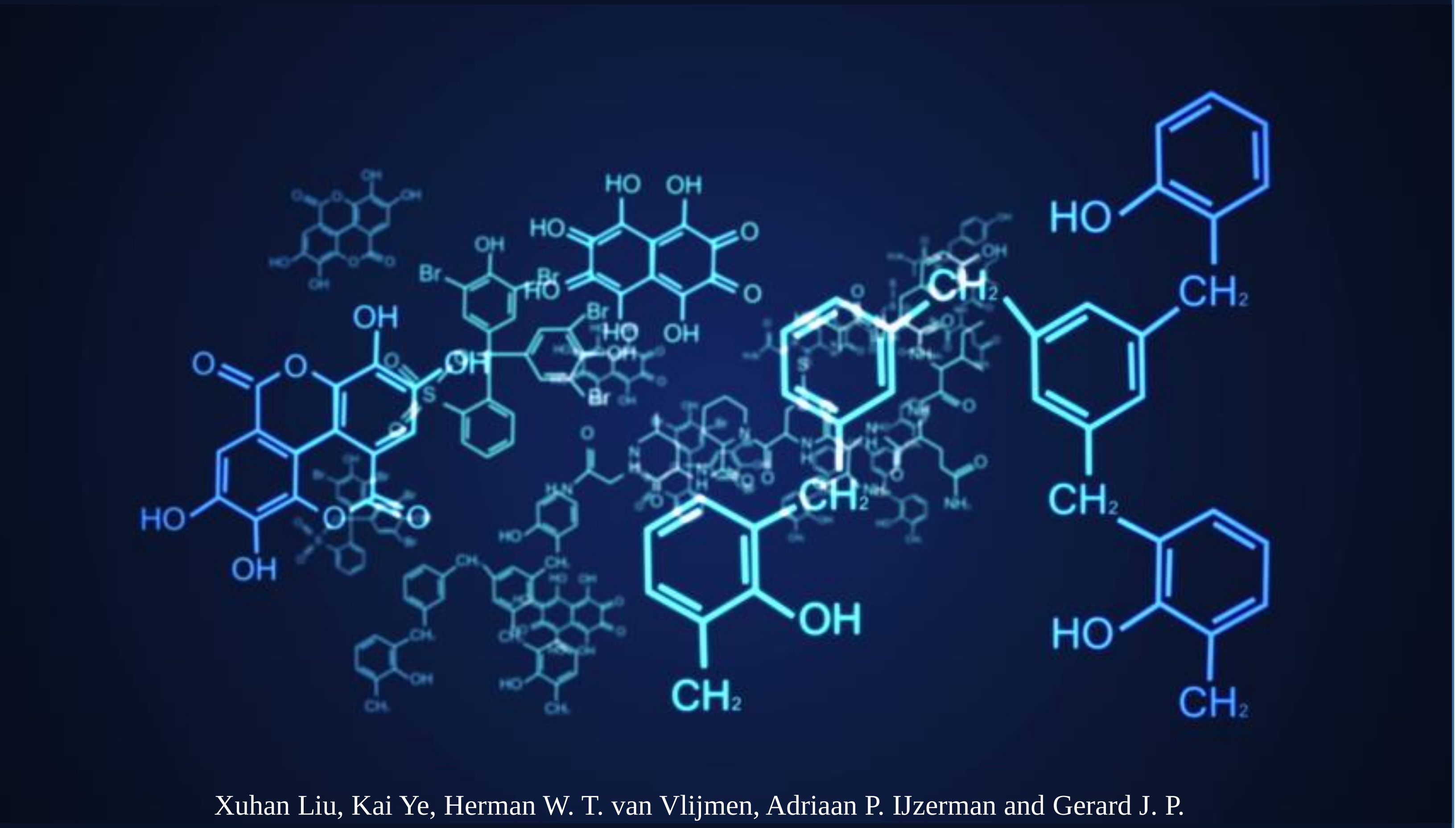


Fig. S4.1: the distribution of SA score and QED score of desired ligand in the *LIGAND* set and molecules generated by *DrugEx v2* with different *ε* in the multi-target case (A-D) and target-specific case (E-H) by using PR (A, B, E and F) and WS (C, D, G and H) rewarding schemes.

Chapter 5

DrugEx v3: scaffold-constrained drug design with graph Transformer-based reinforcement learning



Xuhan Liu, Kai Ye, Herman W. T. van Vlijmen, Adriaan P. IJzerman and Gerard J. P. van Westen*. Preprint. Https://doi.org/10.26434/chemrxiv-2021-px6kz

Abstract

Due to the large drug-like chemical space available to search for feasible drug-like molecules, rational drug design often starts from the specific scaffold to which side chains/substituents are added or modified. With the rapid growth of the application of deep learning in drug discovery, a variety of effective approaches have been developed for de novo drug design. In previous work, we proposed a method named *DrugEx*, which can be applied in polypharmacology based on multi-objective deep reinforcement learning. However, the previous version is trained under fixed objectives similar to other known methods and does not allow users to input any prior information. In order to improve the general applicability, we updated *DrugEx* to design drug molecules based on the scaffold which can contain multiple fragments provided by users. In this work, the Transformer model was employed to generate the structure of molecules. The Transformer is a multi-head self-attention deep learning model containing an encoder for receiving scaffolds as input and a decoder generating molecules as output. In order to deal with the graph representation of molecules, we proposed a novel positional encoding for each atom and bond based on an adjacency matrix to extend the architecture of the Transformer. Each molecule was generated by growing and connecting procedures for the fragments in the given scaffold that were unified into one model. Moreover, we trained this generator under a reinforcement learning framework to increase the number of desired ligands. As a proof of concept, our proposed method was applied to design ligands for the adenosine A_{2A} receptor (A_{2A}AR) and compared it with SMILES-based methods. The results demonstrated its effectiveness in that 100% of generated molecules are valid and most of them had high predicted affinity value towards A_{2A}AR with given scaffold.

Keywords: deep learning, reinforcement learning, policy gradient, drug design, Transformer, multi-objective optimization

5.1. Introduction

Due to the large drug-like chemical space (*i.e.* estimated at 10^{33} - 10^{60} organic molecules) [1], it is impossible to screen every corner of it to discover optimal drug candidates, although high-throughput screening (HTS) technology has been improved significantly in recent years [2]. Commonly, the specific scaffolds derived from endogenous substances are taken as a starting point to design analogs after side chains/substituents are added or modified [3]. These fragments are used as 'building blocks' to develop proper drug leads with combinatorial chemistry such as growing, linking and merging [4]. After a promising drug lead has been discovered, it is further optimized by modifying side chains to improve potency and selectivity which in turn can improve safety and tolerability [5].

The adenosine receptors (ARs) belong to a class of rhodopsin-like GPCRs including four subtypes (A_1, A_{2A}, A_{2B} and A_3). Each of them has a unique pharmacological profile, tissue distribution, and effector coupling [6,7]. ARs are ubiquitously distributed throughout the human tissues, and involved in many biological processes and diseases [8]. Because adenosine is the endogenous agonist of ARs, a number of known ligands of the ARs are adenosine analogs and have a common scaffold. Examples include purines, xanthines, triazines, pyrimidines, and the inclusion of a ribose moiety [9]. In scaffold-based rational drug design, it is generally accepted that a chemical space consisting of 10^9 diverse molecules can be sampled with only 10^3 fragments [10].

Based on rapid developments in the last decade, deep learning has achieved a breakthrough in visual recognition, natural language processing, and other data-rich fields [11]. In drug discovery, deep learning methods have also been extensively used for drug *de novo* design [12]. For distribution-directed issues, Gomez-Bombarelli *et al.* implemented variational autoencoders (VAE) to map molecules into a latent space where each point can also be decoded into unique molecules inversely [13]. They used recurrent neural networks (RNNs) to successfully learn SMILES (simplified molecular-input line-entry system) grammar and construct a distribution of molecular libraries [14]. For goal-directed issues, Sanchez-Lengeling *et al.* combined reinforcement learning and generative adversarial networks

(GANs) to develop an approach named *ORGANIC* to design active compounds toward given targets [15]. Olivecrona *et al.* proposed the *REINVENT* algorithm which updated the reinforcement learning with a Bayesian approach and combined RNNs to generate SMILES-based desired molecules [16,17]. Moreover, Lim *et al.* proposed a method for scaffold-based molecular design with a graph generative model [18]. Li et al. also used deep learning to develop a tool named *DeepScaffold* for this issue [19]. Arús‑Pous *et al.* employed RNNs to develop a SMILES−based scaffold decorator for *de novo* drug design [20]. Yang *et al.* used the Transformer model [21] to develop a tool named *SyntaLinker* for automatic fragment linking [22].

In previous studies, we investigated the performance of RNNs and proposed a method named *DrugEx* by integrating reinforcement learning to balance distribution-directed and goal-directed tasks [23]. Furthermore, we updated it with multi-objective reinforcement learning and applied it in polypharmacology [24]. However, the well-trained model cannot receive any input data from users and only reflect the distribution of the desired molecules with fixed conditions. If the objectives are changed, the model needs to be trained again. In this work, we compared different end-to-end deep learning methods and updated the *DrugEx* model to allow users to provide prior information, such as fragments that should occur in the generated molecules. Based on the extensive experience in our group with the A_{2A}AR, we continue to take this target as an example to evaluate the performance of our proposed methods. In the following context, we will discuss the case of scaffold-constrained drug design, *i.e.* the model takes the scaffolds containing multiple fragments as input to generate desired molecules which also can be predicted to be active to A_{2A}AR. All python code for this study is freely available at http://github.com/XuhanLiu/DrugEx.

5.2. Materials and methods

5.2.1. Data source

Chemical compounds were downloaded from ChEMBL using a SMILES notation (version 27) [25]. After data preprocessing implemented by RDKit, which included neutralizing charges, removing metals and small fragments , ~1.7 million molecules remained for model pre-training. These data were reused from the work about *DrugEx v2* (*ChEMBL* set) [24]. In addition, 10,828 ligands and bioactivity data were extracted from ChEMBL to construct the *LIGAND* set containing structures and activities from bioassays towards four human adenosine receptors. The *LIGAND* set was used for fine-tuning the generative model. Molecules with annotated $A_{2A}AR$ activity were used to train a prediction model. If multiple measurements for the same ligands existed, the average pChEMBL value (pX, including pKi, pKd, pIC50 or pEC50) was calculated and duplicate items were removed. In order to judge if the molecule is desired or not, the threshold of affinity was defined as pX = 6.5 to predict if the compound was active (>= 6.5) or inactive (< 6.5).

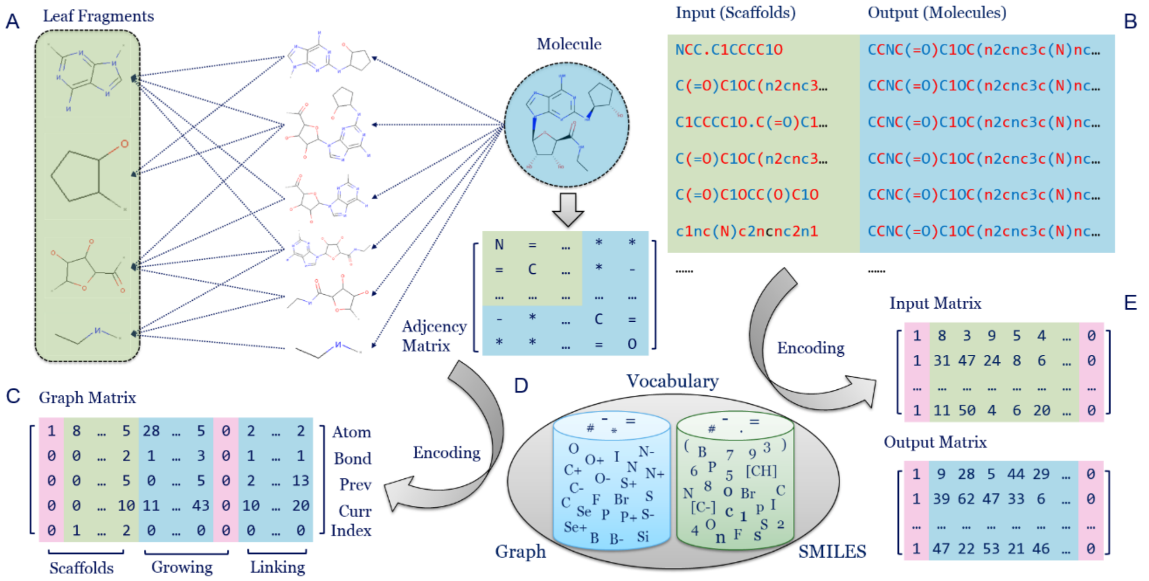


Fig. 5.1: scaffold-molecule pair dataset construction. (A) Each molecule in the dataset is decomposed hierarchically into a series of fragments with the BRICS algorithm. (B) Subsequently data pairs between input and output are created. Combinations of leaf fragments form the scaffold as input, the whole molecule becomes the output. Each token in SMILES sequences is separated by different colors. (C) After conversion to the adjacency matrix, each molecule was represented as a graph matrix. The graph matrix contains five rows, standing for the atom, bond, previous and current positions and fragment index. The columns are composed with three parts to store the information of scaffolds, growing section and linking section. (D) All of tokens are collected to construct the vocabularies for SMILES-based and graph-based generators, respectively. (E) An example of the input and output matrices for the SMILES representation of scaffolds and molecules

Furthermore, the dataset was constructed with an input-output pair for each data point. Each molecule was decomposed into a batch of fragments with BRICS methods [26] in RDKit (Fig. 5.1A). If the molecule contained more than four leaf fragments, the smaller fragments were ignored and a maximum of four larger fragments were reserved to be randomly combined at one time. Here, the scaffold was defined as the combination of different fragments which can be either continuously (linked) or discretely (separated). Their SMILES sequences were joined with ‘.’ as input data which were paired with the full SMILES of molecules. The resulting fragments-molecule pair forms the output data (Fig. 5.1B). After completion of constructing the data pairs, the set was split into a training set and test set with the ratio 9:1 based on the input scaffolds. The resulting *ChEMBL* set contained 10,418,681 and 1,083,271 pairs for training and test set, respectively. The LIGAND set contained 61,413 pairs in the training set and 7,525 pairs in the test set.

5.2.2. Molecular representations

In this study we tested two different molecular representations: SMILES and graph. For SMILES representations each scaffold-molecule pair was transformed into two SMILES sequences which were then split into different tokens to denote atoms, bonds, or other tokens for grammar control (e.g. parentheses or numbers). All of these tokens were put together to form a vocabulary which recorded the index of each token (Fig. 1D). Here, we used the same conversion procedure and vocabulary as in *DrugEx v2*. In addition, we put a start token (GO) at the beginning of a batch of data as input and an end token (END) at the end of the same batch of data as output. After sequence padding with a blank token at empty positions, each SMILES sequence was rewritten as a series of token indices with a fixed length. Subsequently all of these sequences for both scaffolds and molecules were concatenated to construct the input and output matrix (Fig. 1E).

For the graph representation each molecule was represented as a five-row matrix, in which the first two rows stand for the index of the atom and bond types, respectively. The third and fourth rows represent the position of previous and current atoms connected by a bond (Fig. 1C). The columns of this matrix contain three sections to store scaffolds, growing parts, and linking parts. The scaffold section began with a start token in the first row and the last row was labelled the index of each scaffold starting from one. The fragments in the given scaffold for each molecule are put in the beginning of the matrix, followed by the growing part for the scaffold, and the last part is the connection bond between these growing fragments with single bonds. For the growing part and linking sections, the last row was always zero and these two sections were separated by the column of end token. It is worth noticing that the last row was not directly involved in the training process. The vocabulary for graph representation was different from the SMILES format and it contains 38 atom types (Table S5.1) and four bond types (single, double, triple bonds and none). If the atom is the first occurrence in a given scaffold the type of the bond will be empty (indexed as 0 with token '*'). In addition, if the atom at the current position has been recorded in the matrix, the type of the atom will be empty. In order to grasp more details of the graph representation, we also provided the pseudocode for encoding (Table S5.2) and decoding (Table S5.3).

5.2.3. End-to-end deep learning

In this work, we compared three different sequential end-to-end DL architectures to deal with different molecular representations of either graph or SMILES (Fig. 5.2). These methods included: (A) Graph Transformer, (B) LSTM-based encoder-decoder model (LSTM-BASE), (C) LSTM-based encoder-decoder model with attention mechanisms (LSTM+ATTN) and (D) Sequential Transformer model. All of these DL models were constructed with *PyTorch* [27].

For SMILES representation three different models were constructed as follows (Fig. 5.2, right). The encoder and decoder in the LSTM-BASE model (Fig. 5.2B) had the same architectures, containing one embedding layer, three recurrent layers and one output layers (as we did for *DrugEx v2*) [24]. The number of neurons in the embedding and hidden layers were 128 and 512, respectively. The hidden states of the recurrent layer in the encoder are directly sent to the decoder as the initial states. On the basis of LSTM-BASE model, an attention layer was added between the encoder and decoder to form the LSTM+ATTN model (Fig. 5.2C). The attention layer calculates the weight for each position of the input sequence to determine which position the decoder needs to focus on during the decoding process. For each step, the weighted sums of the output calculated by the encoder are combined with the output of the embedding layer in the decoder to form the input for the recurrent layers. The output of the recurrent layers is dealt with by the output layer to generate the probability distribution of tokens in the vocabulary in both of these two models.

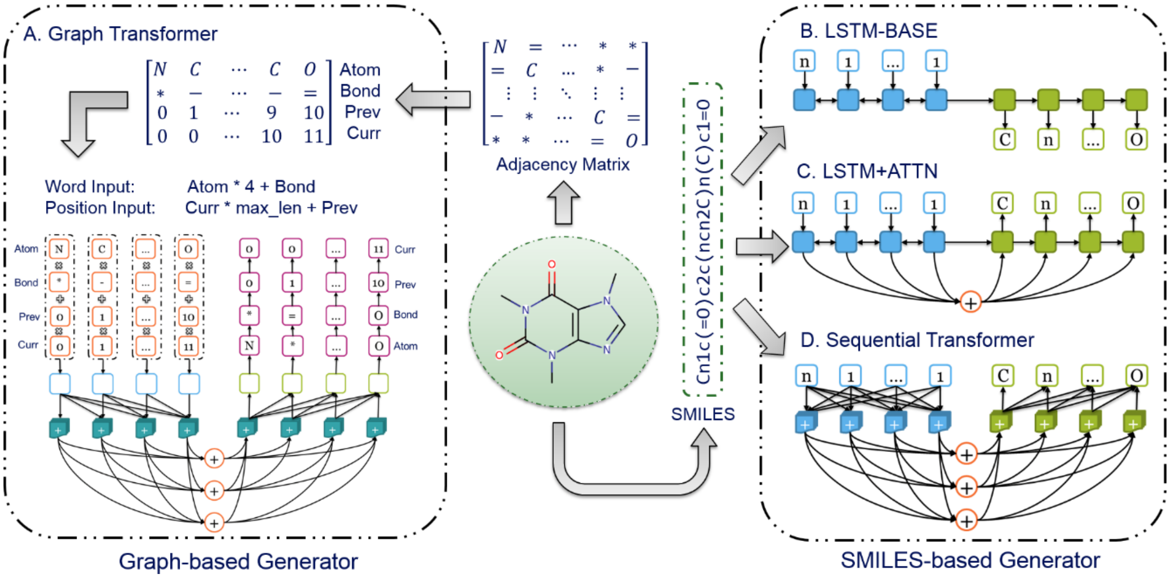


Fig. 5.2: Architectures of four different end-to-end deep learning models: (A) The Graph Transformer; (B) The LSTM-based encoder-decoder model (LSTM-BASE); (C) The LSTM-based encoder-decoder model with attention mechanisms (LSTM+ATTN); (D) The sequential Transformer model. The Graph Transformer accepts a graph representation as input and SMILES sequences are taken as input for the other three models.

The sequential Transformer has a distinct architecture compared to the LSTM+ATTN model although it also exploits an attention mechanism. For the embedding layers "position encodings" are added into the typical embedding structure as the first layer of the encoder and decoder. This ensures that the model no longer needs to encode the input sequence token by token but can process all tokens in parallel. For the position embedding, sine and cosine functions are used to define its formula as follows:

$$PE_{(p,2i)} = \sin(pos/10000^{2i/d_m})$$

$$PE_{(p,2i+1)} = \cos(pos/10000^{2i/d_m})$$

where $PE(p, i)$ is the i^{th} dimension of the position encoding at position p. It has the same dimension $d_m = 512$ as the typical embedding vectors so that the two can be summed.

In addition, self-attention is used in the hidden layers to cope with long-range dependencies.

For each hidden layer in the encoder, it employs a residual connection around a multi-head self-attention sublayer and feed-forward sublayer followed by layer normalization. Besides these two sublayers in the decoder a third sublayer with multi-head attention is inserted to capture the information from output of the encoder.

This self-attention mechanism is defined as the scaled dot-product attention with three vectors: queries (Q), keys (K) and values (V), of which the dimensions are d_q, d_k, d_v, respectively. The output matrix is computed as:

$$Attention(Q,K,V) = softmax\left(\frac{QK^{\top}}{\sqrt{d_k}}\right)V$$

Instead of a single attention function, the Transformer adopts multi-head attention to combine information from different representations at different positions which is defined as:

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(head_1,\ldots,head_h)W^O$$

where h is the number of heads. For each head, the attention values were calculated by different and learned linear projections with Q, K and V as follows:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

where W^O, W^Q, W^K and W^V are metrics of learned weights and we set $h = 8$ as the number of heads and $d_k = d_v = 64$ in this work.

For the graph representation of molecules, we updated the sequential Transformer structure to propose a Graph Transformer (Fig. 5.2A). Similar to the sequential Transformer the Graph Transformer also requires the encodings of both word and position as the input. For the input word, the atom and bond cannot be processed simultaneously; therefore we combined the index of atom and bond together and defined it as follows:

$$I = I_{atom} \times 4 + I_{bond}$$

meaning the index of the input word (I) calculating word vectors are calculated from atom index (I_{atom}) multiplied by four (the total number of bond types defined) and add the bond index (I_{bond}). Similarly, the position of each step cannot be used to calculate the position encoding directly. Faced with more complex data structure than sequential data, Dosovitskiy *et al.* proposed a new positional encoding scheme to define the position for each patch in image data for image recognition [28]. Inspired by their work the position encoding at each step was defined as:

$$P = P_{curr} \times L_{max} + P_{prev}$$

here the input position (P) for calculating the position encoding was the current position (P_{curr}) multiplied by the max length (L_{max}) and adding the previous position (P_{prev}), which was then processed with the same positional encoding method as with the sequential Transformer. For the decoder, the hidden vector from the transformer was taken as the starting point to be decoded by a GRU-based recurrent layer; and the probability of atom, bond, previous and current position was decoded one by one sequentially.

When graph-based molecules are generated, the chemical valence rule is checked in every step. The invalid values of atom and bond types will be masked and an incorrect previous and current position will be removed ensuring the validity of all generated molecules. It is worth noticing that before being encoded, each molecule will be kekulized, meaning that the aromatic rings will be inferred to transform into either single or double bonds. The reason for this is that aromatic bonds interfere with the calculation of the valence value for each atom.

During the training process of SMILES-based models, the negative log likelihood function was used to construct the loss function to guarantee that the token in the output sequence had the largest probability to be chosen. In comparison, the loss function used by the Graph Transformer model also contains four parts for atom, bond, previous and current sites. And the sum of these negative log probability values is minimized to optimize the parameters in the model. For this, the Adam algorithm was used for the optimization of the loss function. Here, the learning rate was set at 10^{-4}, the batch size was 256, and training steps were set to 20 epochs for pre-training and 1,000 epochs for fine-tuning.

5.2.4. Multi-objective optimization

In order to combine multiple objectives we exploited a Pareto-based ranking algorithm with GPU acceleration as mentioned in *DrugEx v2* [24]. Given two solutions m_1 and m_2 with their scores ($x_1, x_2, ..., x_n$) and ($y_1, y_2, ..., y_n$), then m_1 is said to Pareto dominate m_2 if and only if:

$$\forall\, \mathrm{j} \in \{1, \dots, \mathrm{n}\}: x_j \ \geq y_j \ and\ \exists\, \mathrm{j} \in \{1, \dots, \mathrm{n}\}: x_j > y_j$$

otherwise, m_1 and m_2 are non-dominated with each other. After the dominance between all pair of solutions being determined, the non-dominated scoring algorithm is exploited to obtain a rank of Pareto frontiers which consist of a set of solutions. After obtaining frontiers between dominant solutions molecules were ranked based on the average Tanimoto-distance with other molecules instead of crowding distance in the same frontier. Subsequently molecules with smaller distances were ranked on the top. The final reward R^* is defined as:

$$R^* = \begin{cases} 0.5 + \dfrac{k - N_{undesired}}{2N_{desired}}, & if\ desired \\ \dfrac{k}{2N_{undesired}}, & if\ undesired \end{cases}$$

here k is the index of the solution in the Pareto rank and rewards of undesired and desired solutions will be evenly distributed in (0, 0.5] and (0.5, 0.1], respectively.

In this work, we took two objectives into consideration: 1) QED score [29] as implemented by RDKit (from 0 to 1) to evaluate the drug-likeness of each molecule (a larger value means more drug-like) ; 2) an affinity score towards A_{2A}AR which was implemented by a random forest regression model with Scikit-Learn [30] like in *DrugEx v2*. The input descriptors consisted of 2048D ECFP6 fingerprints and 19D physico-chemical descriptors (PhysChem). PhysChem included: molecular weight, logP, number of H bond acceptors and donors, number of rotatable bonds, number of amide bonds, number of bridge head atoms, number of hetero atoms, number of spiro atoms, number of heavy atoms, the fraction of SP3 hybridized carbon atoms, number of aliphatic rings, number of saturated rings, number of total rings, number of aromatic rings, number of heterocycles, number of valence electrons, polar surface area, and Wildman-Crippen MR value. Again it was determined if generated molecules are desired based on the Affinity score (larger than the threshold = 6.5). In addition, the SA score was also exploited to evaluate the synthesizability of generated molecules, which is also calculated by RDKit [31].

5.2.5. Reinforcement learning

In this work we constructed a reinforcement learning framework based on the interplay between the Graph Transformer (agent) and the two scoring functions (environment). A policy gradient method was implemented to train the reinforcement learning model, the objective function is designated as follows:

$$J(\theta) = \mathbb{E}[R^*(y_{1:T})|\theta] = \sum_{t=1}^{T} logG(y_t|y_{1:t-1}) \cdot R^*(y_{1:T})$$

here for each step t during the generation process, the generator (G) determines the probability of each token (y_t) from the vocabulary to be chosen based on the generated sequence in previous steps ($y_{1:t-1}$). In the sequence-based models y_t can only be a token in the vocabulary to construct SMILES while it can be different type of atoms or bonds or the previous or current position in the graph-based model. The parameters in is objective function are updated by employing a policy gradient based on the expected end reward (R^*) received from the predictor. By maximizing this function the parameter θ in the generator can be optimized to ensure that the generator designs desired molecules which obtain a high reward score.

In order to improve the diversity and reliability of generated molecules, we implemented our exploration strategy for molecule generation during the training loops. In the training loop our generator is trained to produce the chemical space as defined by the target of interest. In this strategy there are two networks with the same architectures, an exploitation net (G_θ) and an exploration net (G_φ). G_φ did not need to be trained and its parameters were always fixed and it is based on the general drug-like chemical space for diverse targets obtained from ChEMBL. The parameters in G_θ on the other hand were updated for each epoch based on the policy gradient. Again an *exploring rate* (ε) was defined with a range of [0.0, 1.0] to determine the percentage of scaffolds being randomly selected as input by G_φ to generate molecules. Conversely G_θ generated molecules with other input scaffolds. After the training process was finished G_φ was removed and only G_θ was left as the final model for molecule generation.

5.2.6. Performance evaluation

In order to evaluate the performance of the generators, four coefficients were calculated from the population of generated molecules (validity, accuracy, desirability, and uniqueness) which are defined as:

$$\text{Validity} = \frac{N_{valid}}{N_{total}}$$

$$\text{Accuracy} = \frac{N_{accurate}}{N_{total}}$$

$$\text{Desirability} = \frac{N_{desired}}{N_{total}}$$

$$\text{Uniqueness} = \frac{N_{unique}}{N_{total}}$$

here N_{total} is the total number of molecules, N_{valid} is the number of molecules parsed as valid SMILES sequences, $N_{accurate}$ is the number of molecules that contained given scaffolds, $N_{desired}$ is the number of desired molecules that reach all required objectives, and N_{unique} is the number of molecules which are different from others in the dataset .

To measure molecular diversity, we adopted the Solow Polasky measurement as in *DrugEx v2* [24]. This approach was proposed by Solow and Polasky in 1994 to estimate the diversity of a biological population in an eco-system [32]. The formula to calculate diversity was redefined to normalize the range of values from [1, m] to (0, m] as follows:

$$I(A) = \frac{1}{|A|}\boldsymbol{e}^{\intercal}F(\boldsymbol{s})^{-1}\boldsymbol{e}$$

where A is a set of drug molecules with a size of $|A|$ equal to m, $\boldsymbol{e}$ is an m-vector of 1's and $F(\boldsymbol{s}) = [f(d_{ij}))]$ is a non-singular $m \times m$ distance matrix, in which $f(d_{ij})$ stands for the distance function of each pair of molecule provided as follows:

$$f(d) = e^{-\theta d_{ij}}$$

here we defined the distance d_{ij} of molecules s_i and s_j by using the Tanimoto-distance with ECFP6 fingerprints as follows:

$$d_{ij} = d(s_i, s_j) = 1 - \frac{|s_i \cap s_j|}{|s_i \cup s_j|},$$

where $|s_i \cap s_j|$ represents the number of common fingerprint bits, and $|s_i \cup s_j|$ is the number of union fingerprint bits.

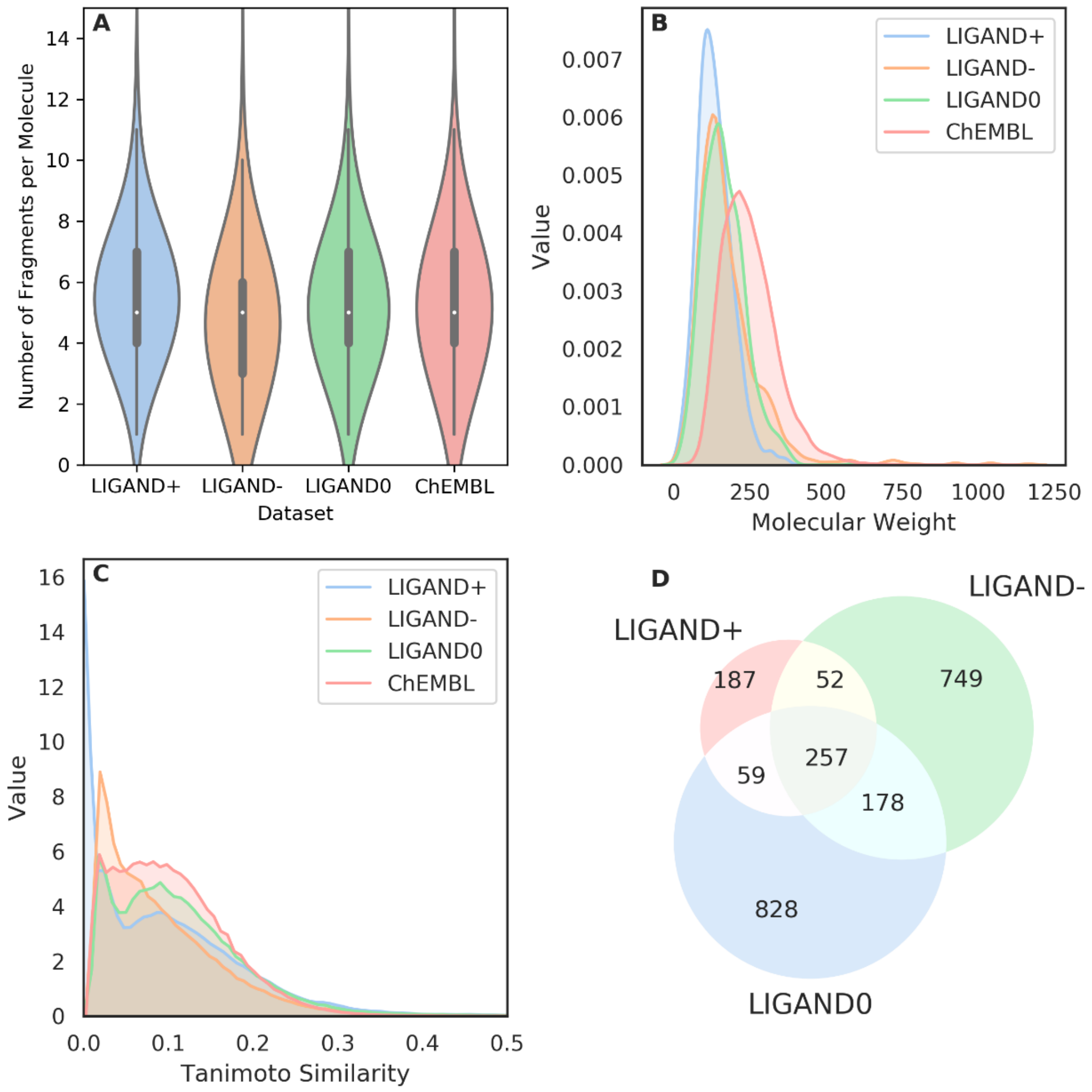


Fig 5.3: Analysis of some properties of fragments in the *ChEMBL* set and three *LIGAND* subsets. (A) Violin plot for the distribution of the number of fragments per molecules; (B) Distribution of molecular weight of these fragments; (C) Distribution of the similarity of the fragments measured by the Tanimoto-similarity with ECFP4 fingerprints; (D) Venn diagram for the intersection of the fragments existing in the three subsets of the *LIGAND* set.

5.3. Results and discussion

5.3.1. Fragmentation of molecule

As stated we decomposed each molecule into a series of fragments with the BRICS algorithm to construct scaffold-molecule pairs. Within BRICS each organic compound can be split into retrosynthetically interesting chemical substructures with a compiled elaborate set of rules. For the *ChEMBL* and *LIGAND* sets, we respectively obtained 194,782 and 2,223 fragments. We further split the LIGAND set into three parts: active ligands (*LIGAND*$^+$, 2,638), inactive ligands (*LIGAND*$^-$, 2710) and undetermined ligands (*LIGAND*^{0}, 5480) based on the pX of bioactivity for A_{2A}AR. The number of fragments in these four datasets have a similar distribution and there are approximately five fragments on average for each molecule with a 95% confidence between [0, 11] (Fig. 5.3A).

In the *LIGAND* set the three subsets have a similar molecular weight distribution of the fragments (Fig. 5.3B) while the average is 164.3Da, smaller than in the *ChEMBL* set (247.3Da). In order to check the similarity of these fragments we used the Tanimoto similarity calculation with ECFP4 fingerprints between each pair of fragments in the same dataset. We found that most of them were smaller than 0.5 indicating that they are dissimilar to each other (Fig. 5.3C). Especially, the fragments in the *LIGAND*$^+$ set have the largest diversity. Moreover, the distribution of different fragments in these three subsets of the *LIGAND* set are shown in Fig. 5.3D. The molecules in these three subsets have their unique fragments and share some common substructures.

5.3.2. Pre-training & fine-tuning

After finishing the dataset construction, four models were pre-trained on the *ChEMBL* set and fine-tuned on the *LIGAND* set. Here, these models were benchmarked on a server with four GTX1080Ti GPUs. After the training process converged each fragment in the test set was presented as input for 10 times to generate molecules. The performance is shown in Table 5.1. The training of Transformer models was faster but consumed more computational resources than LSTM-based methods. In addition, Transformer methods outperformed LSTM-based methods using SMILES. Although the three SMILES-based models improved after being fine-tuned they were still outperformed by the Graph Transformer because of the advantages of the graph representation. To further check the accuracy of generated molecules we also compared the chemical space between the generated molecules and the compounds in the training set with three different representations 1) MW ~ logP; 2) PCA with 19D PhysChem descriptors; 3) tSNE with 2048D ECFP6 fingerprints (Fig. 5.4). The region occupied by molecules generated by the Graph Transformer overlapped completely with the compounds in both the *ChEMBL* and *LIGAND* sets.

Table 5.1: The performance of four different generators for pre-training and fine-tuning processes.

Methods	**Pre-trained Model**		**Fine-tuned Model**		**Time**	**Memory**
	Validity	Accuracy	Validity	Accuracy		
Graph Transformer	100%	99.3%	100%	99.2%	453.8 s	14.5 GB
Sequential Transformer	96.7%	72.0%	99.3%	95.7%	832.3 s	31.7 GB
LSTM-BASE	93.9%	44.1%	98.7%	91.8%	834.6 s	5.5 GB
LSTM+ATTN	89.7%	52.2%	96.4%	90.2%	1212.5 s	15.9 GB

The graph representation for molecules has more advantages over the SMILES representation when dealing with fragment-based molecule design: 1) **Invariance in the local scale**: During the process of molecule generation multiple fragment in the given scaffold can be put into any position in the output matrix without changing the order of atoms and bonds in that fragment. 2) **Extendibility in the global scale**: When the fragments in the scaffold are growing or being linked, they can be flexibly appended in the end column of the graph matrix while the original data structure does not need changing. 3) **Free of grammar**: Unlike in SMILES sequences there is no explicit grammar to constrain the generation of molecules, such as the parentheses for branches and the numbers for rings in SMILES; 4) **Accessibility of chemical rules**: For each added atom or bond the algorithm can detect if the valence of atoms is valid or not and mask invalid atoms or bonds in the vocabulary to guarantee the whole generated matrix can be successfully parsed into a molecule. With these advantages the Graph Transformer generates molecules faster while using less memory.

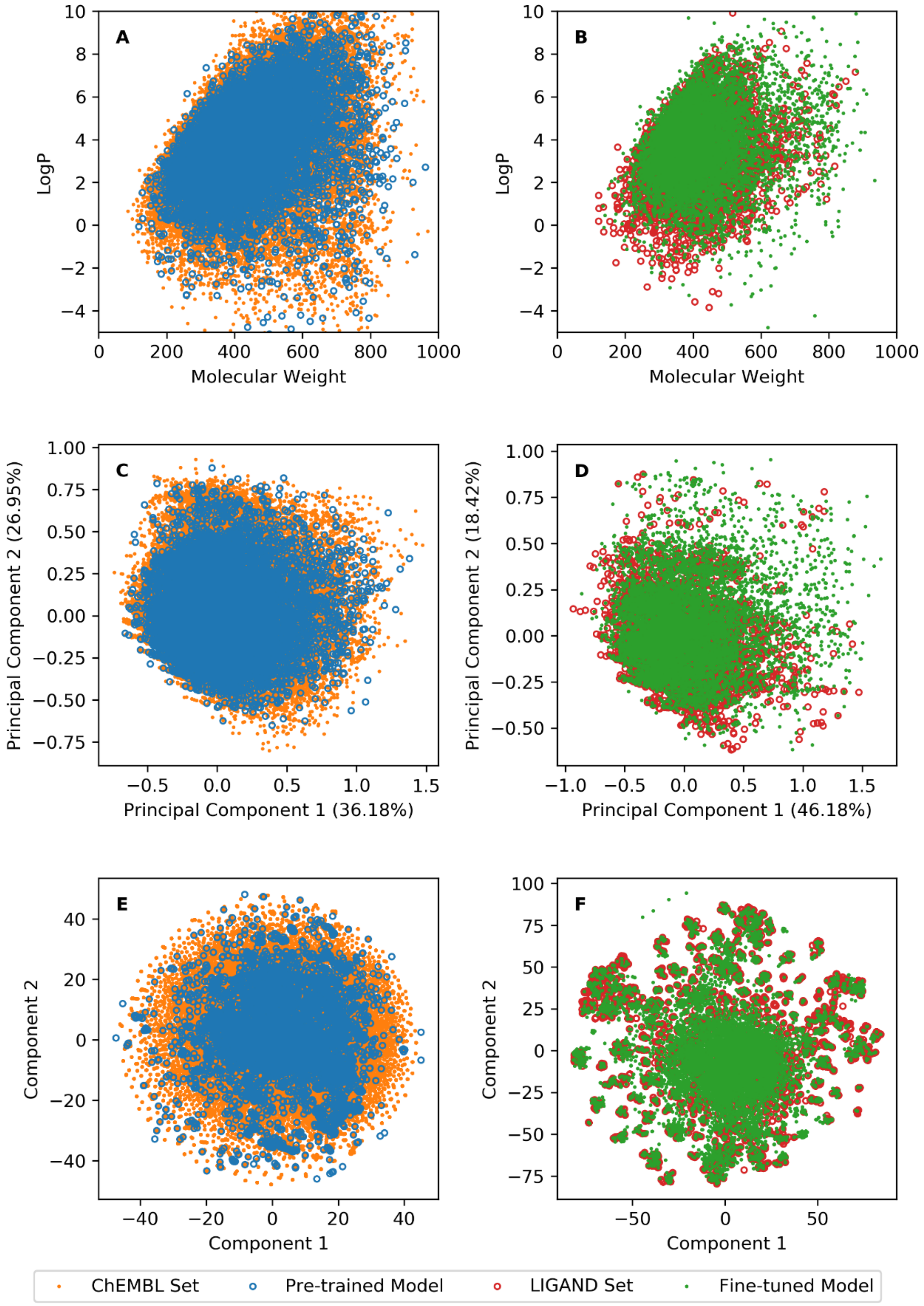


Fig. 5.4: The chemical space of generated molecules by the Graph Transformer pre-trained on the *ChEMBL* set (A, C and E) and being fine-tuned on the *LIGAND* set (B, D and F). Chemical space was represented by either logP ~ MW (A, B) and first two components in PCA on PhysChem descriptors (C, D) and t-SNE on ECFP6 fingerprints (E, F).

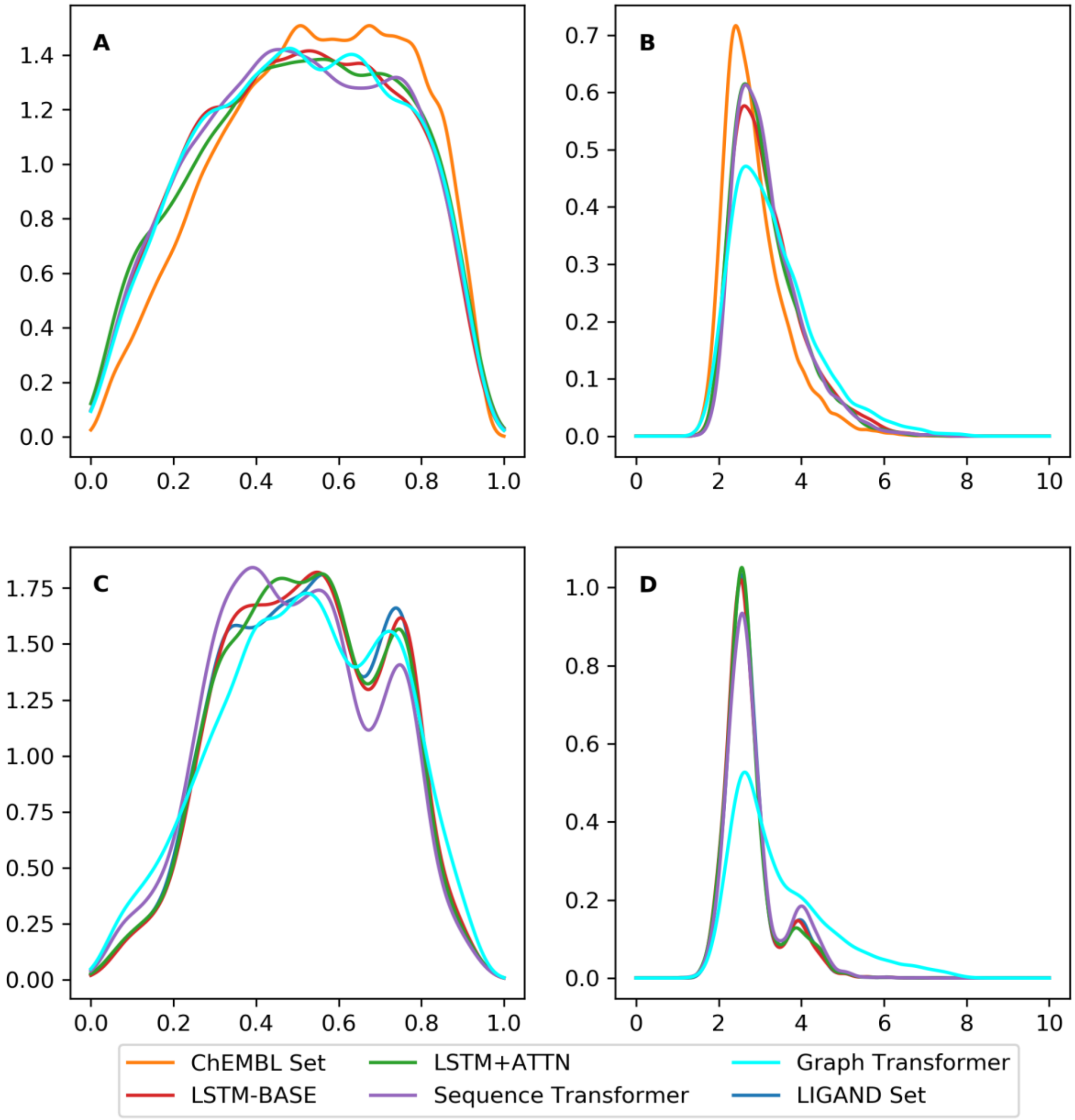


Fig. 5.5: the distribution of QED score (A, C) and SA score (B, D) of desired ligands in the *LIGAND* set and of molecules generated by four different generators.

However, after examining the QED scores and SA scores, we found that although the distribution of QED scores was similar to each other, the synthesizability of the molecules generated by the Graph Transformer were no better than the SMILES-based generators, especially when fine-tuning on the *LIGAND* set (Fig. 5.5). The possible reason is that the molecules generated by the Graph Transformer contains some uncommon rings when the model dealt with long-distance dependencies. In addition, because of more complicated data structure and more parameters in the model, the synthesizability performance of Graph Transformer was not considered high enough when being trained on the small dataset (e.g. the *LIGAND* set). It is also worth noticing that there still was a small fraction of generated molecules that did not contain the given scaffolds. This is caused by the kekulization problem. For example, a scaffold ‘CCC’ can be grown into ‘C1=C(C)C=CC=C1’. After being sanitized, it can be transformed into ‘c1c(C)cccc1’. In this process one single bond in the scaffold is changed to an aromatic bond, which causes the mismatch between the scaffold and the molecule. Currently our algorithm cannot solve this problem because if the aromatic bond is taken into consideration, the valence of aromatic atoms is difficult to be calculated accurately. This would lead to the generation of invalid molecules. Therefore, there is no aromatic bond provided in the vocabulary and all of the aromatic rings are inferred automatically through the molecule sanitization method in RDKit.

5.3.3. Policy gradient

Because the Graph Transformer generates molecules accurately and fast it was chosen as the agent in the RL framework. Two objectives were tested in the training process of this work. The first one was affinity towards $A_{2A}AR$, which is predicted by the random forest-based regression model from *DrugEx v2*; the second one was the QED score calculated with RDKit to measure how similar the generated molecule is to known approved drugs. With the policy gradient method as the reinforcement learning framework two cases were tested. On the one hand, predicted affinity for $A_{2A}AR$ was considered without the QED score. On the other hand, both objectives were used to optimize the model with Pareto ranking. In the first case 86.1% of the generated molecules were predicted active, while the percentage of predicted active molecules in the second case was 74.6%. Although the generator generated more active ligands without the QED score constraint most of them are not drug-like as they always have a molecular weight larger than 500Da. However, when we checked the chemical space represented by tSNE with ECFP6 fingerprints the overlap region between generated molecules and ligands in the training set was not complete implying that they fall out of the applicability domain of the regression model.

In the version of *v2*, we provided an exploration strategy which simulated the idea of evolutionary algorithms such as *crossover* and *mutation* manipulations [24]. However, when coupled to the Graph Transformer there were some difficulties and we had to give up this strategy. Firstly, the mutation strategy did not improve with different mutation rates. A possible reason is that before being generated part the molecule was fixed with a given scaffold, counteracting the effect of mutation caused by the mutation net. Secondly, the *crossover* strategy is computationally very expensive in this context. This strategy needs the convergence of model training and iteratively updates the parameters in the agent. With multiple iterations, it takes a long period of time beyond the computational resources we can currently access. As a result, we updated the exploration strategy as mentioned in the Methods section with six different exploration rates: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5].

Table 5.2: the performance of the Graph Transformer with different exploration rates in the RL framework.

ε	Accuracy	Desirability	Uniqueness	Diversity
0.0	99.7%	74.6%	60.7%	0.879
0.1	99.7%	66.8%	75.0%	0.842
0.2	99.8%	61.6%	80.2%	0.879
0.3	99.7%	56.8%	89.8%	0.874
0.4	99.7%	54.8%	88.8%	0.859
0.5	99.7%	46.8%	88.5%	0.875

Changes to the exploration rate do not influence accuracy and have a low effect on diversity. However desirability (finding active ligands) and uniqueness can be influenced significantly. Empirically determining an optimal value for a given chemical space is recommended.

After training of the models, the scaffolds in the test set were input 10 times to generate molecules. The results for accuracy, desirability, uniqueness, and diversity with different exploration rates are shown in Table 5.2. With a low ε the model generates more desired molecules, but the uniqueness of the generated molecules can be improved. At ε = 0.3 the model generated the highest percentage of unique desired molecules (56.8%). Diversity was always larger than 0.84 and the model achieved the largest value (0.88) with ε = 0.0 or ε = 0.2. The chemical space represented by tSNE with ECFP6 fingerprints confirms that our exploration strategy produces a set of generated molecules completely covering the region occupied by the *LIGAND* set (Fig. 5.6).

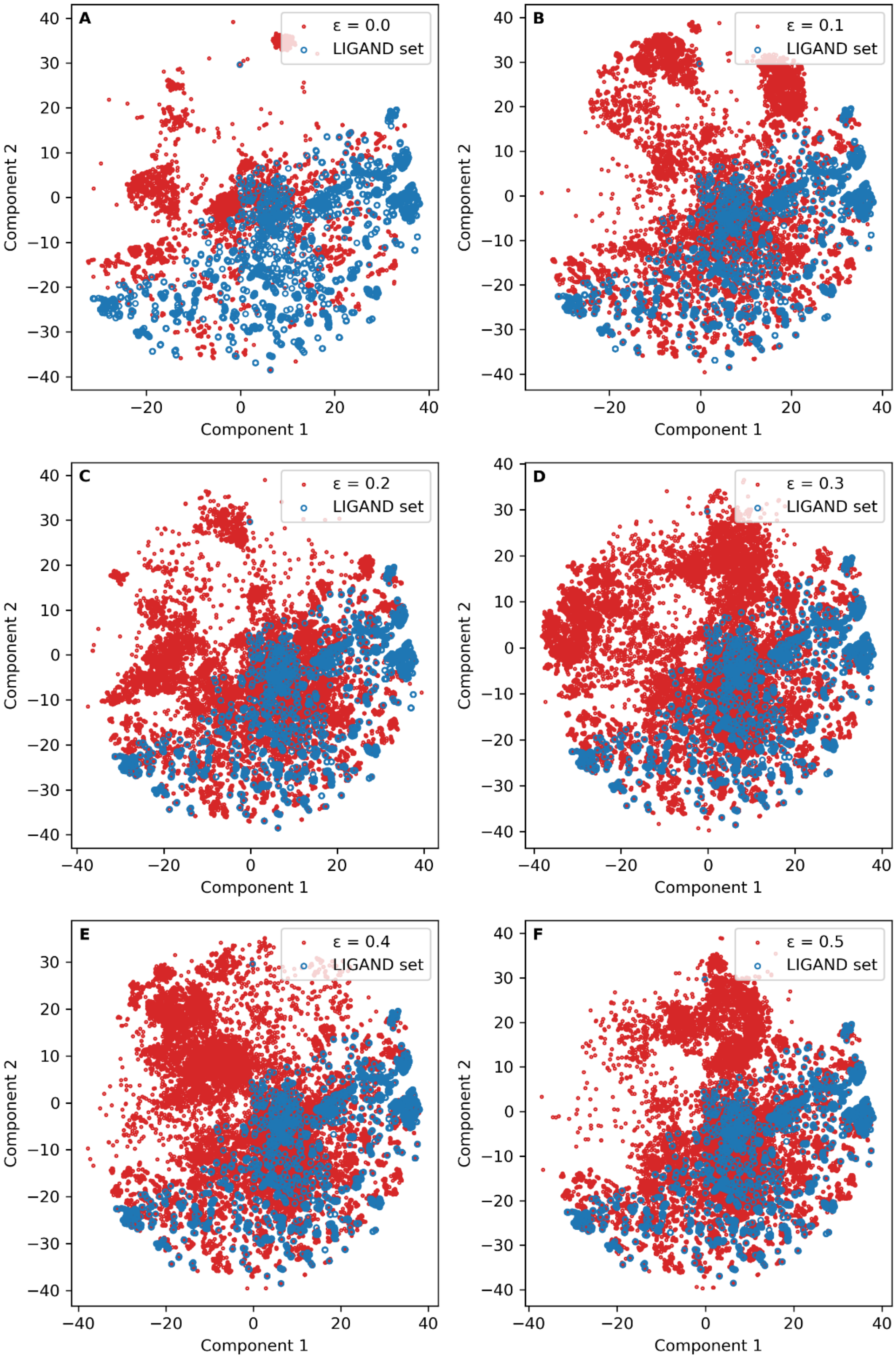


Fig. 5.6: The chemical space of generated molecules by the Graph Transformer trained with different exploration rates in the RL framework. The chemical space was represented by t-SNE on ECFP6 fingerprints.

5.3.4. Generated molecules

In the chemical space for antagonists of A_{2A}AR, furan, triazine, aminotriazole, and purine derivatives such as xanthine and azapurine are common fragments. The Graph Transformer model produced active ligands for A_{2A}AR (inferred from the predictors) with different combinations of these fragments as the scaffolds. Taking these molecules generated by the Graph Transformer as an example, we filtered out the molecules with potentially reactive groups (such as aldehydes) and uncommon ring systems and listed 30 desired molecules as putative A_{2A}AR ligands/antagonists (Fig. 5.7). For each scaffold, five molecules were selected and assigned in the same row. These molecules are considered a valid starting point for further considerations and work (e.g. molecular docking or simulation).

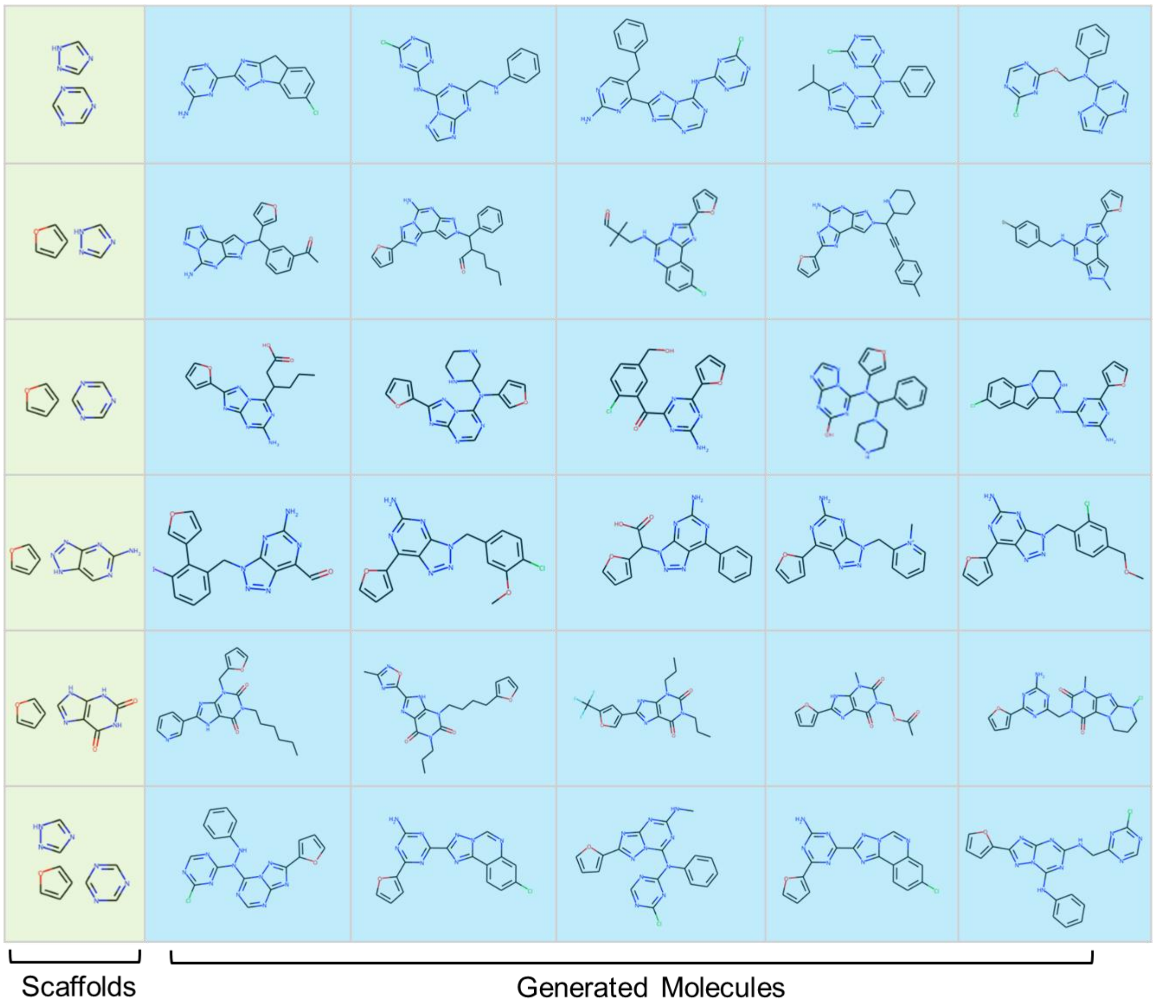


Fig. 5.7: Sample of generated molecules with the Graph Transformer with different scaffolds. These scaffolds include: furan, triazine, aminotriazole, xanthine and azapurine. The generated molecules based on the same scaffolds are aligned in the same row.

5.4. Conclusion and Future Perspective

In this study, *DrugEx* was updated with the ability to design novel molecules based on the scaffolds containing multiple fragments as input. In this version (*v3*), a new positional encoding scheme for atoms and bonds was proposed to make the Transformer model deal with a molecular graph representation. With one model multiple fragments in the scaffold can be grown at the same time and connected to generate a new molecule. In addition, chemical rules on valence are enforced at each step of the process of molecule generation to ensure that all generated molecules are valid. This is impossible for SMILES-based generation, as SMILES-based molecules are constrained by grammar that allows a 2D topology to be represented in a sequential way. With multi-objective reinforcement learning the model generates drug-like ligands, in our case for the A_{2A}AR target.

In future work, the Graph Transformer will be extended to include other information as input to design drugs conditionally. For example, proteochemometric modelling (PCM) can take information for both ligands and targets as input to predict the affinity of their interactions, which allows promiscuous (useful for e.g., viral mutants) or selective (useful for e.g., kinase inhibitors) properties [33]. The Transformer can then be used to construct inverse PCM models which take the protein information as input (e.g. sequences, structures or descriptors) to design active ligands for a given protein target without known ligands. Moreover, the Transformer can also be used for lead optimization. For instance, the input can be a “hit” already, generating “optimized” ligands, or a “lead” with side effects to produce ligands with a better ADME/tox profile.

Declarations

Availability of data and materials

The data used in this study is publicly available ChEMBL data, the algorithm published in this manuscript is made available at https://github.com/XuhanLiu/DrugEx.

Authors’ Contributions

XL and GJPvW conceived the study and performed the experimental work and analysis. KY, APIJ nd HWTvV provided feedback and critical input. All authors read, commented on and approved the final manuscript.

Acknowledgements

XL thanks Chinese Scholarship Council (CSC) for funding, GJPvW thanks the Dutch Research Council and Stichting Technologie Wetenschappen (STW) for financial support (STW-Veni #14410). Thanks go to Dr. Xue Yang for verifying Table S1 and Dr. Anthe Janssen checking the convergence of tSNE. We also acknowledge Bert Beerkens for providing the common scaffolds used to generate molecules as an example.

Competing Interests

The authors declare that they have no competing interests

References

1. Polishchuk PG, Madzhidov TI, Varnek A (2013) Estimation of the size of drug-like chemical space based on GDB-17 data. J Comput Aided Mol Des 27 (8):675-679. doi:10.1007/s10822-013-9672-4
2. Hajduk PJ, Greer J (2007) A decade of fragment-based drug design: strategic advances and lessons learned. Nat Rev Drug Discov 6 (3):211-219. doi:10.1038/nrd2220
3. Card GL, Blasdel L, England BP, Zhang C, Suzuki Y, Gillette S, Fong D, Ibrahim PN, Artis DR, Bollag G, Milburn MV, Kim SH, Schlessinger J, Zhang KY (2005) A family of phosphodiesterase inhibitors discovered by cocrystallography and scaffold-based drug design. Nat Biotechnol 23 (2):201-207. doi:10.1038/nbt1059
4. Bian Y, Xie XS (2018) Computational Fragment-Based Drug Design: Current Trends, Strategies, and Applications. AAPS J 20 (3):59. doi:10.1208/s12248-018-0216-7
5. Hughes JP, Rees S, Kalindjian SB, Philpott KL (2011) Principles of early drug discovery. Br J Pharmacol 162 (6):1239-1249. doi:10.1111/j.1476-5381.2010.01127.x
6. Fredholm BB (2010) Adenosine receptors as drug targets. Exp Cell Res 316 (8):1284-1288. doi:10.1016/j.yexcr.2010.02.004
7. Chen JF, Eltzschig HK, Fredholm BB (2013) Adenosine receptors as drug targets--what are the challenges? Nat Rev Drug Discov 12 (4):265-286. doi:10.1038/nrd3955
8. Moro S, Gao ZG, Jacobson KA, Spalluto G (2006) Progress in the pursuit of therapeutic adenosine receptor antagonists. Med Res Rev 26 (2):131-159. doi:10.1002/med.20048
9. Jespers W, Oliveira A, Prieto-Diaz R, Majellaro M, Aqvist J, Sotelo E, Gutierrez-de-Teran H (2017) Structure-Based Design of Potent and Selective Ligands at the Four Adenosine Receptors. Molecules 22 (11). doi:10.3390/molecules22111945
10. Sheng C, Zhang W (2013) Fragment informatics and computational fragment-based drug design: an overview and update. Med Res Rev 33 (3):554-598. doi:10.1002/med.21255
11. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521 (7553):436-444. doi:10.1038/nature14539
12. Liu X, IJzerman AP, van Westen GJP (2021) Computational Approaches for De Novo Drug Design: Past, Present, and Future. Methods Mol Biol 2190:139-165. doi:10.1007/978-1-0716-0826-5_6
13. Gomez-Bombarelli R, Wei JN, Duvenaud D, Hernandez-Lobato JM, Sanchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel TD, Adams RP, Aspuru-Guzik A (2018) Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. ACS Cent Sci 4 (2):268-276. doi:10.1021/acscentsci.7b00572
14. Segler MHS, Kogej T, Tyrchan C, Waller MP (2018) Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. ACS Cent Sci 4 (1):120-131. doi:10.1021/acscentsci.7b00512
15. Benjamin S-L, Carlos O, Gabriel L. G, Alan A-G (2017) Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). doi:10.26434/chemrxiv.5309668.v3
16. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. Journal of cheminformatics 9 (1):48. doi:10.1186/s13321-017-0235-x
17. Blaschke T, Arus-Pous J, Chen H, Margreitter C, Tyrchan C, Engkvist O, Papadopoulos K, Patronov A (2020) REINVENT 2.0: An AI Tool for De Novo Drug Design. Journal of chemical information and modeling 60 (12):5918-5922. doi:10.1021/acs.jcim.0c00915
18. Lim J, Hwang SY, Moon S, Kim S, Kim WY (2019) Scaffold-based molecular design with a graph generative model. Chem Sci 11 (4):1153-1164. doi:10.1039/c9sc04503a

19. Li Y, Hu J, Wang Y, Zhou J, Zhang L, Liu Z (2020) DeepScaffold: A Comprehensive Tool for Scaffold-Based De Novo Drug Discovery Using Deep Learning. Journal of chemical information and modeling 60 (1):77-91. doi:10.1021/acs.jcim.9b00727
20. Arus-Pous J, Patronov A, Bjerrum EJ, Tyrchan C, Reymond JL, Chen H, Engkvist O (2020) SMILES-based deep generative scaffold decorator for de-novo drug design. Journal of cheminformatics 12 (1):38. doi:10.1186/s13321-020-00441-8
21. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin IJae-p (2017) Attention Is All You Need.arXiv:1706.03762
22. Yang Y, Zheng S, Su S, Zhao C, Xu J, Chen H (2020) SyntaLinker: automatic fragment linking with deep conditional transformer neural networks. Chem Sci 11 (31):8312-8322. doi:10.1039/d0sc03126g
23. Liu X, Ye K, van Vlijmen HWT, IJzerman AP, van Westen GJP (2019) An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine A2A receptor. Journal of cheminformatics 11 (1):35. doi:10.1186/s13321-019-0355-6
24. Liu X, Ye K, van Vlijmen HWT, Emmerich M, IJzerman AP, van Westen GJP (2021) DrugEx v2: de novo design of drug molecules by Pareto-based multi-objective reinforcement learning in polypharmacology. Journal of cheminformatics 13 (1):85. doi: 10.1186/s13321-021-00561-9
25. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, Light Y, McGlinchey S, Michalovich D, Al-Lazikani B, Overington JP (2012) ChEMBL: a large-scale bioactivity database for drug discovery. Nucleic Acids Res 40 (Database issue):D1100-1107. doi:10.1093/nar/gkr777
26. Degen J, Wegscheid-Gerlach C, Zaliani A, Rarey M (2008) On the art of compiling and using 'drug-like' chemical fragment spaces. ChemMedChem 3 (10):1503-1507. doi:10.1002/cmdc.200800178
27. PyTorch. https://pytorch.org/.
28. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby NJae-p (2020) An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.arXiv:2010.11929
29. Bickerton GR, Paolini GV, Besnard J, Muresan S, Hopkins AL (2012) Quantifying the chemical beauty of drugs. Nat Chem 4 (2):90-98. doi:10.1038/nchem.1243
30. Scikit-Learn: machine learning in Python. http://www.scikit-learn.org/.
31. Ertl P, Schuffenhauer A (2009) Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. Journal of cheminformatics 1 (1):8. doi:10.1186/1758-2946-1-8
32. Solow AR, Polasky S (1994) Measuring biological diversity. Environmental and Ecological Statistics 1 (2):95-103. doi:10.1007/BF02426650
33. van Westen GJ, Wegner JK, Geluykens P, Kwanten L, Vereycken I, Peeters A, Ijzerman AP, van Vlijmen HW, Bender A (2011) Which compound to select in lead optimization? Prospectively validated proteochemometric models guide preclinical development. PLoS One 6 (11):e27518. doi:10.1371/journal.pone.0027518

Table S5.1: Atoms in vocabulary for graph-based molecule generation.

Symbol	Valence	Charge	Number	Word
O	2	0	8	2O
O+	3	1	8	3O+
O-	1	-1	8	1O-
C	4	0	6	4C
C+	3	1	6	3C+
C-	3	-1	6	3C-
N	3	0	7	3N
N+	4	1	7	4N+
N-	2	-1	7	2N-
Cl	1	0	17	1Cl
S	2	0	16	2S
S	6	0	16	6S
S	4	0	16	4S
S+	3	1	16	3S+
S+	5	1	16	5S+
S-	1	-1	16	1S-
F	1	0	9	1F
I	1	0	53	1I
I	5	0	53	5I
I+	2	1	53	2I+
Br	1	0	35	1Br
P	5	0	15	5P
P	3	0	15	3P
P+	4	1	15	4P+
Se	2	0	34	2Se
Se	6	0	34	6Se
Se	4	0	34	4Se
Se+	3	1	34	3Se+
Si	4	0	14	4Si
B	3	0	5	3B
B-	4	-1	5	4B-
As	5	0	33	5As
As	3	0	33	3As
As+	4	1	33	4As+
Te	2	0	52	2Te
Te	4	0	52	4Te
Te+	3	1	52	3Te+
*	0	0	0	*

The column of ‘Symbol’ is the symbol of the atom and its charge; the column of ‘Valence’ is the value of valence of the state of each chemical element; the ‘Number’ column stands for the index of each element in the periodic table, the last row is the unique word for each state of these elements, a combination of its valence and symbol.

Table S5.2: The pseudo code for encoding the graph representation of molecules in *DrugEx v3*

```
Algorithm encoding:
    Input:
        mol: structure of the kekulized molecule
        subs: structure of the scaffolds
        vocab: vocabulary of tokens which is consisted of graph matrix
    Output:
         matrix: the n x 5 matrix to represents the molecular graph.

    # Ensure the atom of the subs are put at the start in the molecule
    mol ← RANK_ATOM_BY_SUB(mol, subs)
    sub_atoms ← GET_ATOMS (subs)
    sub_bonds ← GET_BONDS (subs)
    mol_atoms ← GET_ATOMS (mol)
    frag, grow, link ← [('GO', 0, 0, 0, 1)], [], [(0, 0, 0, 0, 0)]
    For atom in mol_atoms:
        # The bonds which connect to the atom having the index before this atom
        bonds ← GET_LEFT_BONDS (mol, atom)
        For bond in bonds:
            tk_bond ← GET_TOKEN (vocab, bond)
            other ← GET_OTHER_ATOM(mol, atom, bond)
            If IS_FIRST (bonds, bond):
                tk_atom ← GET_TOKEN (vocab, atom)
            Else:
                tk_atom ← GET_TOKEN (vocab, None)

            # The index of the scaffold in which the current atom locates
            # Its value starts from 1. If it is not in the scaffold, it will be 0
            scf ← GET_FRAG_ID (subs, atom)
            column ← (tk_atom, tk_bond, GET_INDEX (other), GET_INDEX (atom), scf)
            If other in sub_atoms and atom in sub_atoms and bond not in sub_bonds:
                Insert column to link
            Else if bond in sub_bonds:
                Insert column to frag
            Else:
                Insert column to grow
        End
    End
    Insert ('EOS', 0, 0, 0, 0) to grow
    matrix ← CONCATENATE_BY_COLUMN (frag, grow, link)
    Return matrix
```

Table S5.3: The pseudo code for decoding the graph representation of molecules in *DrugEx v3*

```
Algorithm decoding:
    Input:
       matrix: the n x 5 matrix to represents the molecular graph
       vocab: vocabulary of tokens which is consisted of graph matrix
    Output:
       mol: structure of the kekulized molecule
       subs: structure of the scaffolds

     mol ← new MOL ()
     subs ← new SUB ()
     For atom, bond, prev, curr, scf in matrix:
         If atom == 'EOS' or atom == 'GO':
             continue
         If atom != '*':
             a ← new Atom (GET_ATOM_SYMBOL(vocab, atom))
             SET_FORMAL_CHARGE (a, GET_CHARGE(vocab, atom))
             ADD_ATOM (mol, a)
             If scf != 0: ADD_ATOM (subs, a)
         If bond != 0:
             b ← new Bond (bond)
             ADD_BOND(mol, b)
             If frag != 0:
                 ADD_BOND (subs, b)
     End

     # automatically determine the aromatic rings
     mol ← SANITIZE (mol)
     subs ← SANITIZE (subs)
    Return mol, subs
```

Chapter 6

GenUI: interactive and extensible open source software platform for *de novo* molecular generation and cheminformatics

M. Sicho[†], X. Liu[†], D. Svozil & G. J. P. van Westen[*]. Journal of Cheminformatics (2021).

Https://doi.org/10.1186/s13321-021-00550-y

Abstract

Many contemporary cheminformatics methods, including computer-aided *de novo* drug design, hold promise to significantly accelerate and reduce the cost of drug discovery. Thanks to this attractive outlook, the field has thrived and in the past few years has seen an especially significant growth, mainly due to the emergence of novel methods based on deep neural networks. This growth is also apparent in the development of novel *de novo* drug design methods with many new generative algorithms now available. However, widespread adoption of new generative techniques in the fields like medicinal chemistry or chemical biology is still lagging behind the most recent developments. Upon taking a closer look, this fact is not surprising since in order to successfully integrate the most recent *de novo* drug design methods in existing processes and pipelines, a close collaboration between diverse groups of experimental and theoretical scientists needs to be established. Therefore, to accelerate the adoption of both modern and traditional *de novo* molecular generators, we developed GenUI (Generator User Interface), a software platform that makes it possible to integrate molecular generators within a feature-rich graphical user interface that is easy to use by experts of diverse backgrounds. GenUI is implemented as a web service and its interfaces offer access to cheminformatics tools for data preprocessing, model building, molecule generation, and interactive chemical space visualization. Moreover, the platform is easy to extend with customizable frontend React.js components and backend Python extensions. GenUI is open source and a recently developed *de novo* molecular generator, DrugEx, was integrated as a proof of principle. In this work, we present the architecture and implementation details of GenUI and discuss how it can facilitate collaboration in the disparate communities interested in *de novo* molecular generation and computer-aided drug discovery.

Keywords: graphical user interface, de novo drug design, molecule generation, deep learning, web application

t

6.1. Introduction

Due to significant technological advances in the past decades, the body of knowledge on the effects and roles of small molecules in living organisms has grown tremendously [1, 2]. At present, we assume the number of entries across all databases to be in the range of hundreds of millions or billions (10^8-10^9) [3-5] and a large portion of this data has also accumulated in public databases such as ChEMBL [6, 7] or PubChem BioAssay [1]. However, the size of contemporary databases is still rather small when compared to some estimates of the theoretical size of the drug-like chemical space, which may contain up to 10^{33} unique structures according to a recent study [8]. However, it should be noted that numerous studies in the past reported numbers both bigger and smaller depending on the definition used [8-11]. In addition, considering that only 1-2 measured biological activities per compound are available [12], the characterization of known compounds also needs to be expanded.

For a long time, *de novo* drug design algorithms for systematic and rational exploration of chemical space [13-15] and quantitative structure-activity relationship (QSAR) modeling [16] have been considered promising and useful cheminformatics tools to efficiently broaden our horizons with less experimental costs and without the need to exhaustively evaluate as many as 10^{33} possible drug-like compounds to find the few of interest. The relevance of QSAR modeling and *de novo* compound design for drug discovery has been discussed many times [13-21], but these approaches can be just as useful in other research areas [16]. In chemical biology, new tool compounds and chemical probes can be discovered with these methods as well [22].

Thanks to the rapid growth of bioactivity databases and widespread utilization of graphical processing units (GPUs) the efforts to develop powerful data-driven approaches for *de novo* compound generation and QSAR modeling based on deep neural networks (DNNs) has grown substantially (Fig. 6.1) [19]. The most attractive feature of DNNs for *de novo* drug design is their ability to probabilistically generate compound structures [13, 23]. DNNs are able to take non-trivial structure-activity patterns into account, thereby increasing the potential for scaffold hopping and the diversity of designed molecules [24, 25]. A number of generators based on DNNs was developed recently demonstrating the ability of various network architectures to generate compounds of given properties [13, 23, 26-29]. However, it should also be noted that the field of *de novo* drug design and molecular generation also has a long history of evolutionary heuristic methods with genetic algorithms on the forefront [20]. These traditional methods are still being investigated and developed [30-35] and it is yet to be established how they compare to the novel DNN-based approaches [13].

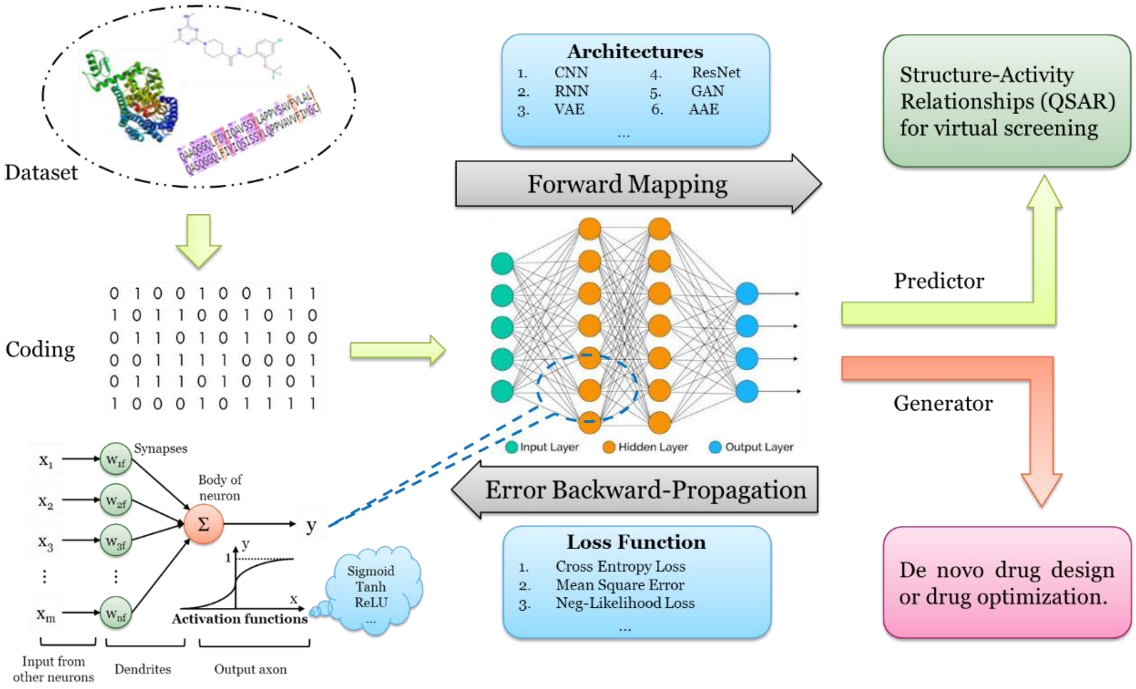


Figure 6.1: Schematic view of a typical cheminformatics workflow involving a DNN. First, a data set of compound structures and their measured activities on the desired target molecule (most often a protein) is compiled and encoded to suitable representation. Second, the encoded data is used as input of the neural network in forward mapping. A large number of architectures can be used with recurrent neural networks (RNNs) and convolutional neural networks (CNNs) as the most popular examples. Finally, the neural network is trained by back-propagating the error of a suitable loss function to adjust the activations inside the network so that the loss is minimized. Depending on the architecture, the network is trained either as a bioactivity predictor (e.g. a QSAR model) or as a molecular generator.

Although *de novo* molecular design algorithms have been in development for multiple decades [36] and experimentally validated active compounds have been proposed [18, 37-45], these success stories are still far away from the envisaged performance of the ‘robot scientist’ [46-48]. Successful development of a completely automated and sufficiently accurate and efficient closed loop process has been elusive, but significant advances have been made nonetheless [45]. However, even with encouraging results suggesting that full automation of the drug discovery process might be possible [18, 45, 49-52], human insight and manual labor are still necessary to further refine and evaluate the compounds generated by *de novo* drug design algorithms. In particular, human intervention is of utmost importance in the process of compound scoring whereby best candidates are prioritized for synthesis and experimental validation [18, 51]. In this instance, the contributions of artificial intelligence (AI) are significant and AI algorithms can work independently to some extent, but expert knowledge is still important to interpret and refine such results and the creation of comprehensive graphical user interfaces (GUIs) and interoperable software packages can facilitate more direct involvement of experts from various fields.

Though many *in silico* compound generation and optimization tools are available for free [53], it is still an exception that these approaches are routinely used since the vast majority of methods described in the literature serve only as a proof of concept and can rarely be considered production-ready software. In particular, they lack a proper GUI through which non-experts could easily access the algorithms and analyze their inputs and outputs in a convenient way. While there are many notable exceptions [33, 35, 54-58], the implemented GUIs are often simplistic and intended to be used only with one particular method. In addition, many molecular generators would also benefit from a comprehensive and easy to use application programming interface (API) that would enable easier integration with existing computational tools and infrastructures. Recently an open source tool called Flame was presented that offers many of the aforementioned features in the field of predictive QSAR modeling [59]. Such integrated frameworks from the realm of *de novo* compound generation are much rarer, however. To the best of our knowledge, BRADSHAW [60] and Chemistry42 [61] are the only two that were disclosed in literature recently and they unfortunately have not been made available as open source, which limits their use by the scientific community. On the other hand, it should be noted that there has been effort to develop open and interactive databases of generated structures as evidenced by the most recent example, cheML.io [62], which allows open access to the generated structures, but does not support “on-the-fly” generation. We argue that the lack of easy to use and auditable information systems for *de novo* drug design is a factor leading to some level of disconnection between medicinal and computational chemists [63], which can stand in the way of effective utilization of many promising *de novo* drug design tools.

Therefore, in this work we present the development of GenUI, a cheminformatics software framework that provides a GUI and APIs for easy use of molecular generators by human experts as well as their integration with existing drug discovery pipelines and other automated processes. The GenUI framework integrates solutions for import, generation, storage and retrieval of compounds, visualization of the created molecular data sets and basic utilities for QSAR modeling. Therefore, it is also suitable for many basic cheminformatics tasks (i.e. visualization of chemical data sets or simple QSAR modeling).

All GenUI features can be easily accessed through the web-based GUI or the REST API (Representational State Transfer API) to ensure that both human users and automated processes can interact with the application with ease. Integration of new molecular generators and other features is facilitated by a documented Python API while quick GUI customization is possible with an extensive library of components implemented with the React.js JavaScript library. To demonstrate the features of the GenUI framework, our recently published molecular generator DrugEx [64] was integrated within the GenUI ecosystem. The source code of the GenUI platform is distributed under the MIT open-source license [65-67] and several Docker [68-70] images are also available online for quick deployment [71].

6.2. Software architecture

User interaction with GenUI happens through the frontend web client which issues REST API calls to the backend, which comprises five services (Fig. 6.1). However, advanced users may also implement clients and automated processes that use the REST API directly.

The five backend services form the core parts of GenUI and can be described as follows:

1. “Projects” service handles user account management, authorization, and workflows. It is used to log users in and organize their work into projects.
2. “Compounds” service manages the compound database including deposition, standardization, and retrieval of molecules and the associated data (i.e. bioactivities, physicochemical properties, or chemical identifiers).
3. “QSAR Models” service facilitates the training and use of QSAR models. They can be used to predict biological activities of the generated compounds, but they are also integral to training of many molecular generators.
4. “Generators” service is responsible for the integration of *de novo* molecular generators. It is meant to be used to set up and train generative algorithms whether they are based on traditional approaches or deep learning.
5. “Maps” service enables the creation of 2D chemical space visualizations and integration of dimensionality reduction algorithms.

In the following sections, the design and implementation of each part of the GenUI platform will be described in more detail.

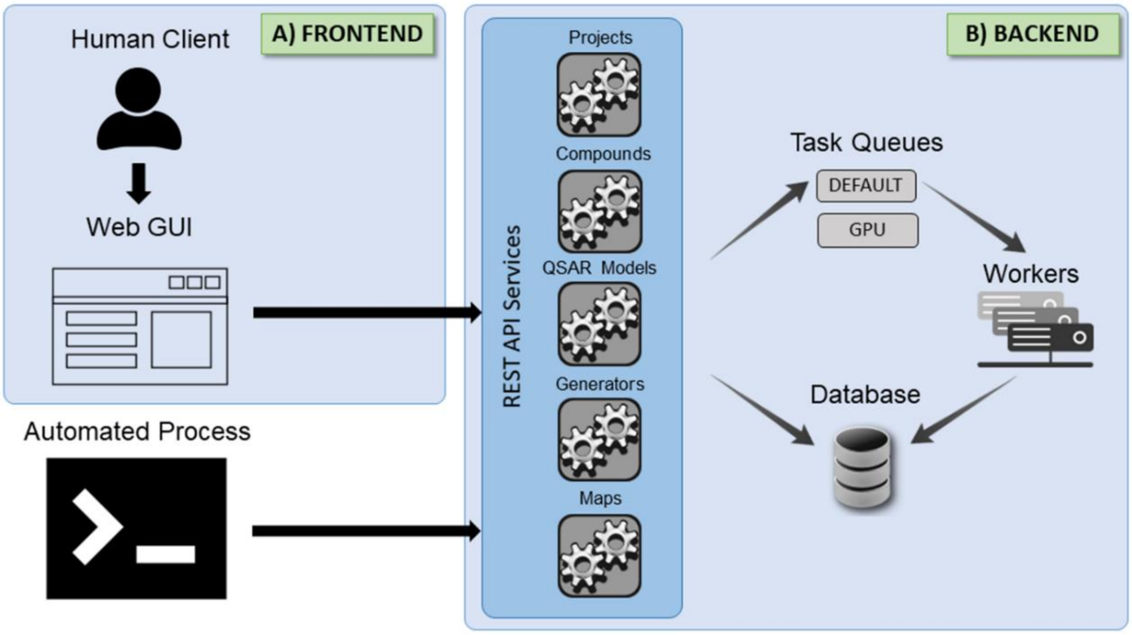


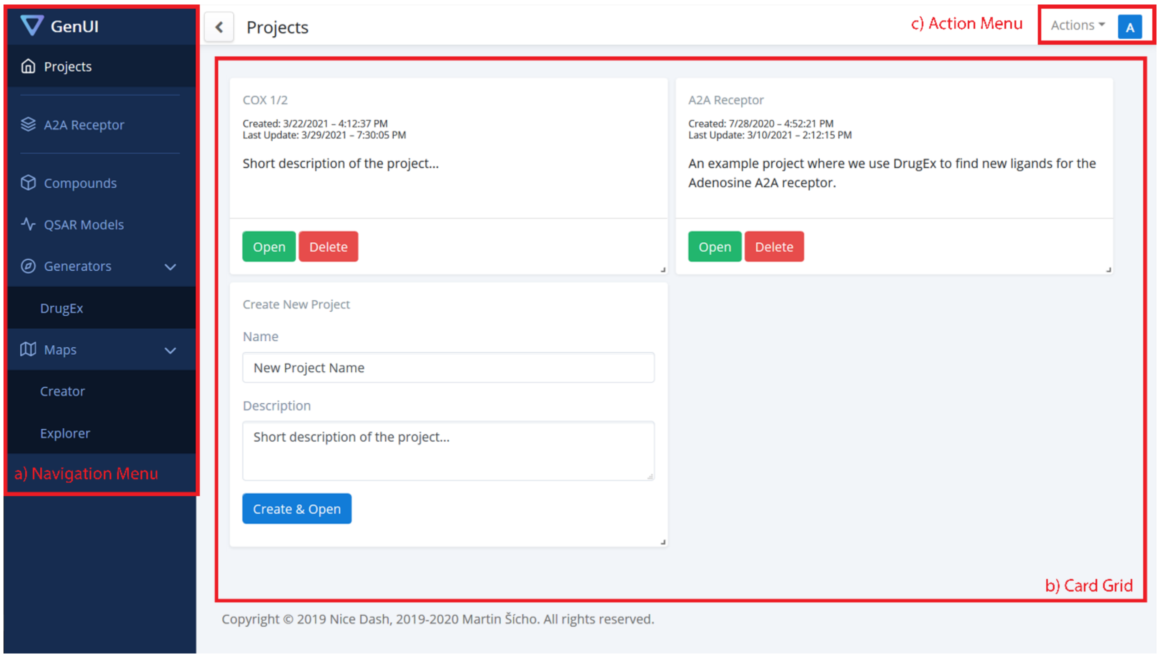
Fig. 6.1 Schematic depiction of the GenUI platform architecture. On the frontend (A), users interact with the web-based GUI to access the backend server REST API services (B). All actions and data exchange are facilitated through REST API calls so that any automated process can also interact with GenUI. The backend application comprises five REST API services each of which has access to the data storage and task queue subsystems. The services can issue computationally intensive and long-running asynchronous tasks to backend workers to ensure sufficient responsiveness and scalability. In

the current implementation, tasks can be submitted to two queues: (1) the default CPU queue, which handles all tasks by default, or (2) the GPU queue, intended for tasks that can be accelerated by the use of GPUs.

6.3. Frontend

6.3.1. Graphical user interface (GUI)

The GUI is implemented as a JavaScript application built on top of the React.js [72] web framework. The majority of graphical components is provided by the Vibe Dashboard open-source project [73], but the original collection of Vibe components was considerably expanded with custom components to fetch, send, and display data exchanged with the GenUI backend. In addition, frameworks Plotly.js [74], Charts.js [75] and ChemSpace.js [76] are used to provide helpful interactive figures.



COX 1/2

Created: 3/22/2021 – 4:12:37 PM
Last Update: 3/29/2021 – 7:30:05 PM

Short description of the project...

A2A Receptor

Created: 7/28/2020 – 4:52:21 PM
Last Update: 3/10/2021 – 2:12:15 PM

An example project where we use DrugEx to find new ligands for the Adenosine A2A receptor.

Create New Project

Name: New Project Name

Description: Short description of the project...

Copyright © 2019 Nice Dash, 2019-2020 Martin Šícho. All rights reserved.

Fig. 6.2: A screenshot showing part of the GenUI web GUI. The GUI is in a state where the "A2A Receptor" project is already open and the navigation menu on the left can be used to access its data. The GUI consists of three main parts: a) navigation menu, b) card grid and c) action menu. The navigation menu is used to browse data associated with various GenUI services ("Projects" in this case). If a link is clicked in the navigation menu, the data of the selected service is displayed as a grid of interactive cards. Each card allows the users to manage particular data items (a project in this case). The action menu in the top right is also updated depending on the service selected in the navigation menu and performs actions not related to a particular data item. In this case, the action menu was used to bring up the project creation form represented by the card in the bottom left of the card grid.

The GUI reflects the structure of the GenUI backend services (Fig. 6.1 and Fig. 6.2). Each backend service (“Projects”, “Compounds”, “QSAR Models”, “Generators”, and “Maps”) is represented as a separate item in the navigation menu on the left side of the interface (Fig. 6.2a). Upon clicking a menu item, the corresponding page opens rendering a grid of cards (Fig. 6.2b) that displays the objects corresponding to the selected backend service. Various actions related to the particular service can be performed from the action menu in the top right of the interface (Fig. 6.2c).

6.3.2. Projects

The “Projects” interface serves as a simple way to organize user workflows. For example, a project can encapsulate a workflow for the generation of novel ligands for one protein target (Fig. 6.2). Each project contains imported compounds, QSAR models, molecular generators and chemical space maps. The number of projects per user is not limited and they can be deleted or created as needed.

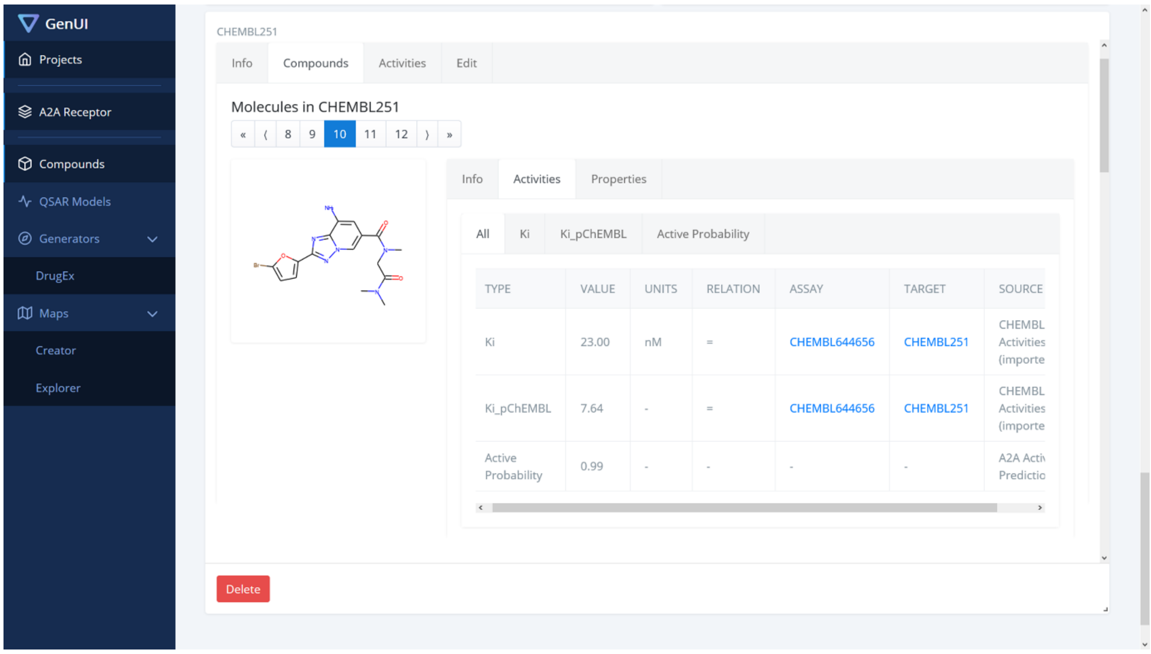
6.3.3. Compounds

Each project may contain any number of compound sets (Fig. 6.3). Each set of compounds can have a different purpose in the project and come from a different source. Therefore, the contents of each card on the card grid depend on the type of compound set the card represents. Compounds can be generated by generators, but also imported from SDF files, CSV files or obtained directly from the ChEMBL database [6, 7]. New import filters can be easily added by extending the Python backend and customizing the components of the React API accordingly (see Python API and JavaScript API). For each compound in the compound set the interface can display its 2D representation (Fig. 6.3), molecular identifiers (i.e. SMILES, InChI, and InChIKey), reported and predicted activities (Fig. 6.3) and physicochemical properties (i.e. molecular weight, number of heavy atoms, number of aromatic rings, hydrogen bond donors, hydrogen bond acceptors, logP and topological polar surface area).

6.3.4. QSAR models

All QSAR models trained or imported in the given project are available from the “QSAR

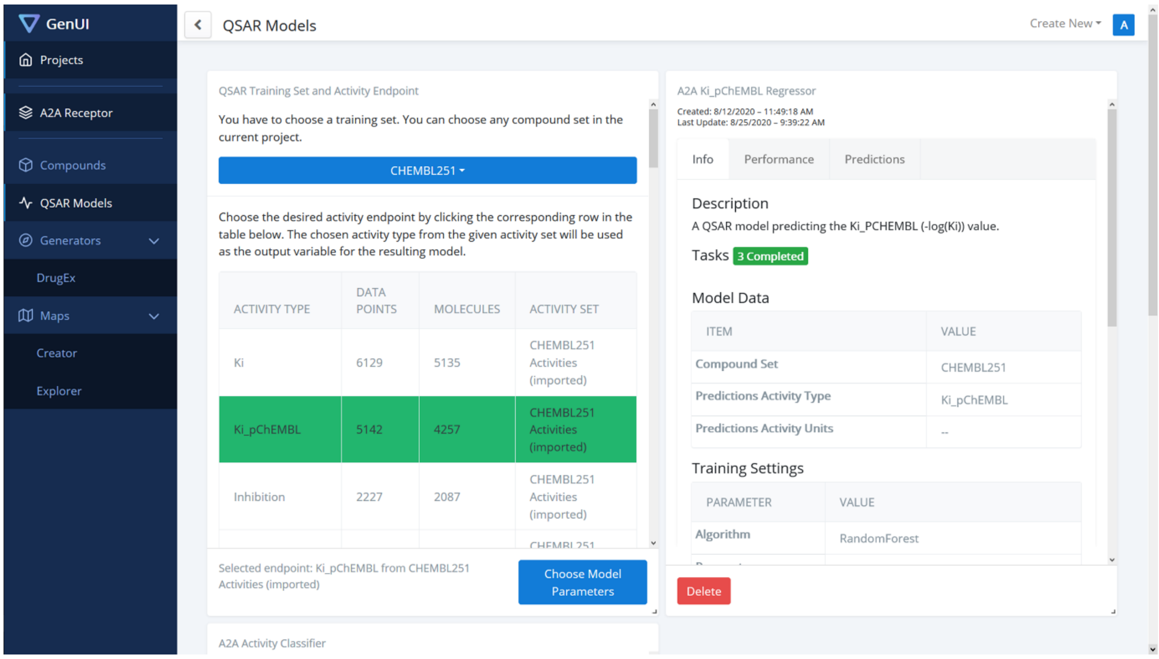
Models” page (Fig. 6.5, Fig. 6.6). Each QSAR model is represented by a card with several tabs. The “Info” tab contains model metadata, as well as a serialized model file to download (Fig. 6.5). The “Performance” tab lists various performance measures of the QSAR model obtained by cross-validation or on an independent hold out test set (Fig. 6.6). The validation procedure can be adjusted by the user during model creation (Fig. 6.5). Making predictions with the model is possible under the “Predictions” tab. Each QSAR model can be used to make predictions for any compound set listed on the “Compounds” page and the calculated predictions will then become visible in that interface as well (Fig. 6.4).



TYPE	VALUE	UNITS	RELATION	ASSAY	TARGET	SOURCE
Ki	23.00	nM	=	CHEMBL644656	CHEMBL251	CHEMBL Activities (importe
Ki_pChEMBL	7.64	-	=	CHEMBL644656	CHEMBL251	CHEMBL Activities (importe
Active Probability	0.99	-	-	-	-	A2A Acti Predictio

Fig. 6.3: A screenshot showing part of the “Compounds” GUI. Users can import data sets from various sources. A card representing an already imported data set from the ChEMBL database [7] is shown. The position and size of each displayed card can be modified by either dragging the card (reposition) or adjusting the bottom right corner (size change). The card shown is currently expanded over two rows of the card grid (Fig. 6.3b) in order to accommodate the displayed data better. The “Activities” tab in the compound overview shows summary of the biological activity data associated with the compound. The activities are grouped by type and aside from experimentally determined activities the interface also displays activity predictions of available QSAR models. For example, in the view shown the “Active Probability” activity type is used to denote the output probability from a classification QSAR model. Each activity value also contains information about its origin (the “Source” column) so that it can be tracked back to its source.

New QSAR models are submitted for training with a creation card (Fig. 6.5) that helps users choose model hyperparameters and a suitable training strategy (i.e. the characteristics of the independent hold out validation set, the number of cross-validation folds or the choice of validation metrics). The "Info" tab of a trained model contains important metadata as well as a hyperlink to export the model and save it as a reusable Python object. This import/export feature enables users to archive and share their work, enhancing the reusability and reproducibility of the developed models [77]. The "Performance" tab can be used to observe model performance data according to the chosen validation scheme (Fig. 6.6). This information is different depending on the chosen model type (regression vs. classification, Fig. 6.6a vs. Fig. 6.6b) and the parameters used (i.e. the choice of validation metrics). Additional performance measures and machine learning algorithms can be integrated with the backend Python API. Creation of such extensions does not even require editing of the GUI for many standard algorithms (see Python API).



QSAR Training Set and Activity Endpoint

You have to choose a training set. You can choose any compound set in the current project.

Choose the desired activity endpoint by clicking the corresponding row in the table below. The chosen activity type from the given activity set will be used as the output variable for the resulting model.

ACTIVITY TYPE	DATA POINTS	MOLECULES	ACTIVITY SET
Ki	6129	5135	CHEMBL251 Activities (imported)
Ki_pChEMBL	5142	4257	CHEMBL251 Activities (imported)
Inhibition	2227	2087	CHEMBL251 Activities (imported)

Selected endpoint: Ki_pChEMBL from CHEMBL251 Activities (imported)

A2A Ki_pChEMBL Regressor

Created: 8/12/2020 – 11:49:18 AM
Last Update: 8/25/2020 – 9:39:22 AM

Description

A QSAR model predicting the Ki_PCHEMBL (-log(Ki)) value.

Tasks 3 Completed

Model Data

ITEM	VALUE
Compound Set	CHEMBL251
Predictions Activity Type	Ki_pChEMBL
Predictions Activity Units	--

Training Settings

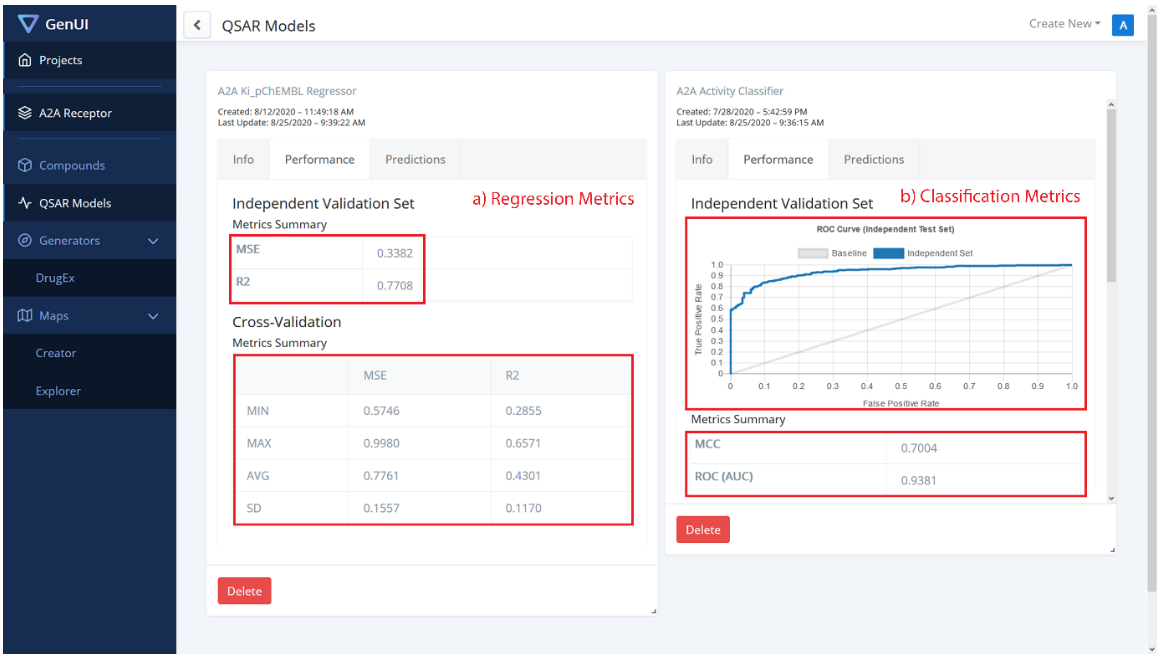
PARAMETER	VALUE
Algorithm	RandomForest

Fig. 6.4: A screenshot showing part of the "QSAR Models" GUI. The card on the left side of the screen shows how training data is chosen for a new model while the card on the right shows metadata about an already trained model.

6.3.5. Generators

Under the "Generators" menu item, the users find a list of individual generators implemented in the GenUI framework (Fig. 6.7). Currently, only the DrugEx generator [64] is available, but other generators can be added by extending the Python backend (see

Python API) and customizing the existing React components (see JavaScript API). It is likely that some generators will have specific requirements on the GUI elements used on the page and, thus, the GUI is organized so that each new generator is integrated as a completely new page accessed from the navigation menu on the left. Many of the graphical elements used in the DrugEx extension (i.e. the model creation form, Fig. 6.7a) are simply customized elements from the library of GenUI graphical components. In fact, the GUI for DrugEx is based on the same React components as the “QSAR Models” view.



A2A Ki_pChEMBL Regressor

Created: 8/12/2020 – 11:49:18 AM
Last Update: 8/25/2020 – 9:39:22 AM

Independent Validation Set

Metrics Summary

MSE	0.3382
R2	0.7708

Cross-Validation

Metrics Summary

	MSE	R2
MIN	0.5746	0.2855
MAX	0.9980	0.6571
AVG	0.7761	0.4301
SD	0.1557	0.1170

A2A Activity Classifier

Created: 7/28/2020 – 5:42:59 PM
Last Update: 8/25/2020 – 9:36:15 AM

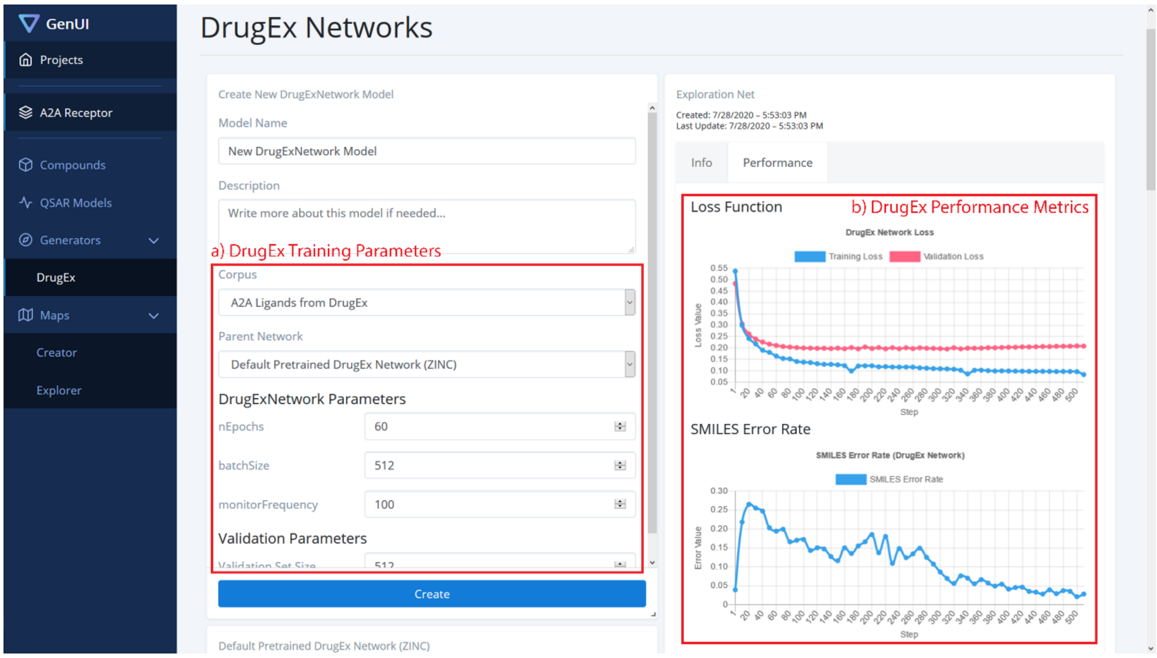
Independent Validation Set

Metrics Summary

MCC	0.7004
ROC (AUC)	0.9381

Fig. 6.5: Performance evaluation view for (a) regression and (b) classification QSAR model. In (a) the mean-squared error (MSE) and the coefficient of determination (R2) are used as validation metrics. In (b) the performance is measured on a holdout independent validation test set with the Matthews correlation coefficient (MCC) and the area under the receiver operating characteristic (ROC) curve (AUC). The ROC curve itself is also displayed above the metrics.

The DrugEx method consists of two networks, an exploitation network and an exploration network, that are trained together [78]. The exploration network is used to fine-tune the exploitation network, which is then trained under the reinforcement learning framework to optimize the agent that generates the desired compounds. Therefore, the interface of DrugEx was divided into two parts: 1) for training DrugEx exploration networks (Fig. 6.7) and 2) for training DrugEx agents (not shown). In this case, the graphical elements needed for the two types of networks are very similar and are just placed as two card grids under each other. The only custom React components made for this interface are the figures used to track real time model performance (Fig. 6.7b). All other components come from the original GenUI React library (see JavaScript API) and are simply configured to use data from the DrugEx extension REST API endpoints.



Create New DrugExNetwork Model

Model Name: New DrugExNetwork Model

Description: Write more about this model if needed...

Corpus: A2A Ligands from DrugEx

Parent Network: Default Pretrained DrugEx Network (ZINC)

DrugExNetwork Parameters

nEpochs: 60

batchSize: 512

monitorFrequency: 100

Validation Parameters

Validation Set Size: 512

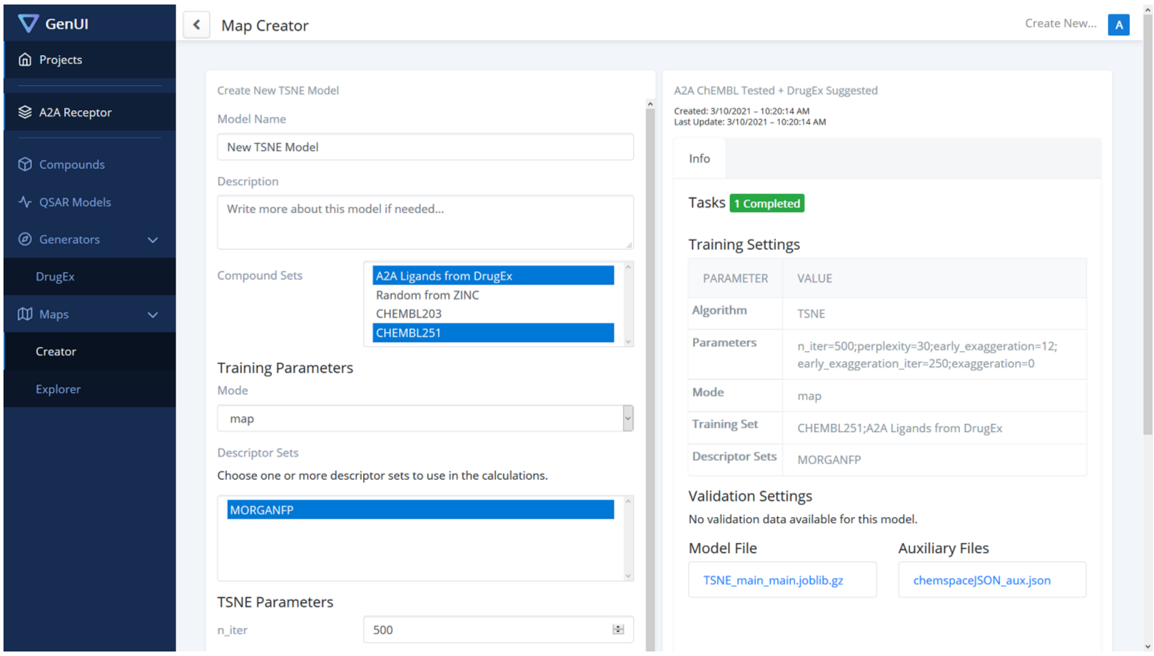
Default Pretrained DrugEx Network (ZINC)

Exploration Net

Created: 7/28/2020 – 5:53:03 PM
Last Update: 7/28/2020 – 5:53:03 PM

Fig. 6.6: A screenshot showing part of the "DrugEx" GUI with a model creation card with (a) DrugEx training parameters and (b) performance overview of a trained DrugEx network. In (a) the fields to define the compound set for the process of fine-tuning the exploitation ('parent') recurrent neural network trained on the ZINC data set [63] are shown. In addition, the form provides fields to set the number of learning epochs, training batch size, frequency of performance monitoring and size of the validation set. In (b) the "Performance" tab tracks model performance. It shows values of the loss function on the training set and validation set (top) and the SMILES error rate (bottom) at each specified step of the training process. The performance view is updated according to the chosen monitoring frequency in real time as the model is being trained. Each model also has the "Info" tab which holds the same information as for QSAR models.

Like QSAR models, DrugEx networks can also be serialized and saved as files. For example, a cheminformatics researcher can build a DrugEx model outside of the GenUI ecosystem (i.e. using the scripts published with the original paper [64]) and provide the created model files to another researcher who can import and use the model from the GenUI web-based GUI. Therefore, it is easy to share work and accommodate various groups of users in this way.



Create New TSNE Model

Model Name: New TSNE Model

Description: Write more about this model if needed...

Compound Sets: A2A Ligands from DrugEx; Random from ZINC; CHEMBL203; CHEMBL251

Training Parameters

Mode: map

Descriptor Sets

Choose one or more descriptor sets to use in the calculations.

MORGANFP

TSNE Parameters

n_iter: 500

A2A ChEMBL Tested + DrugEx Suggested

Created: 3/10/2021 – 10:20:14 AM
Last Update: 3/10/2021 – 10:20:14 AM

Tasks 1 Completed

Training Settings

PARAMETER	VALUE
Algorithm	TSNE
Parameters	n_iter=500;perplexity=30;early_exaggeration=12;early_exaggeration_iter=250;exaggeration=0
Mode	map
Training Set	CHEMBL251;A2A Ligands from DrugEx
Descriptor Sets	MORGANFP

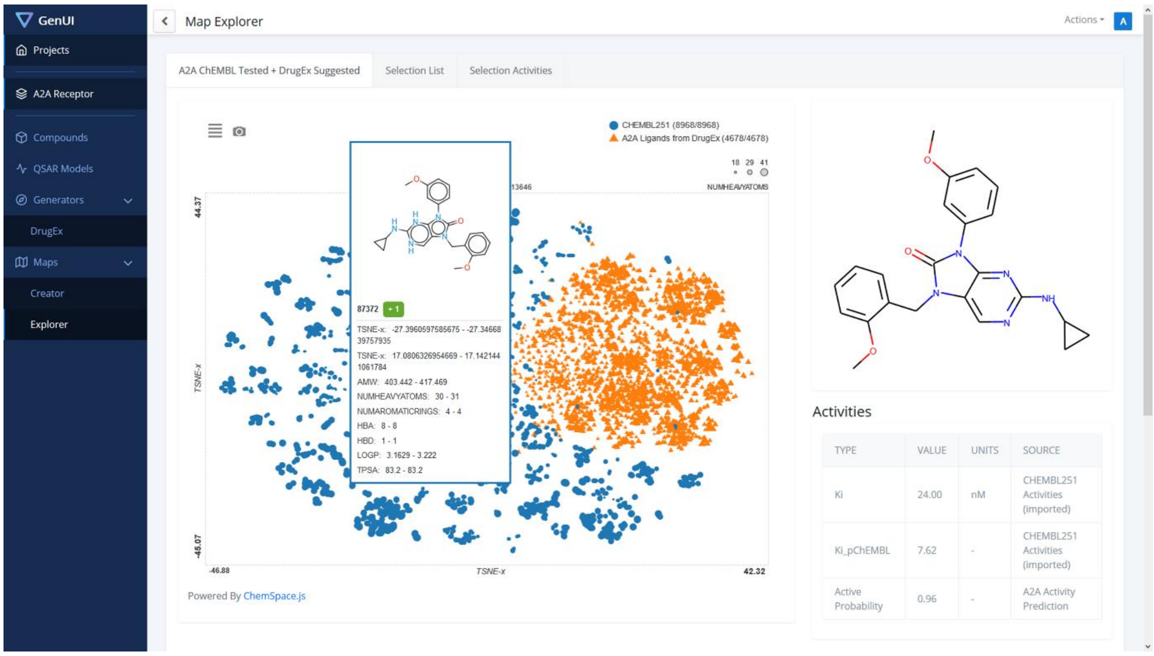
Validation Settings

No validation data available for this model.

Model File: TSNE_main_main.joblib.gz

Auxiliary Files: chemspaceJSON_aux.json

Fig. 6.7: The "Creator" interface of GenUI "Maps" page. On the left a form to create a new t-SNE [78] mapping of two sets of compounds using Morgan fingerprints is shown while information about an existing map can be seen on the right.



87372 +1

TSNE-x: -27.3960597585675 - -27.3466839757935
TSNE-x: 17.0806326954669 - 17.1421441061784
AMW: 403.442 - 417.469
NUMHEAVYATOMS: 30 - 31
NUMAROMATICRINGS: 4 - 4
HBA: 8 - 8
HBD: 1 - 1
LOGP: 3.1629 - 3.222
TPSA: 83.2 - 83.2

Activities

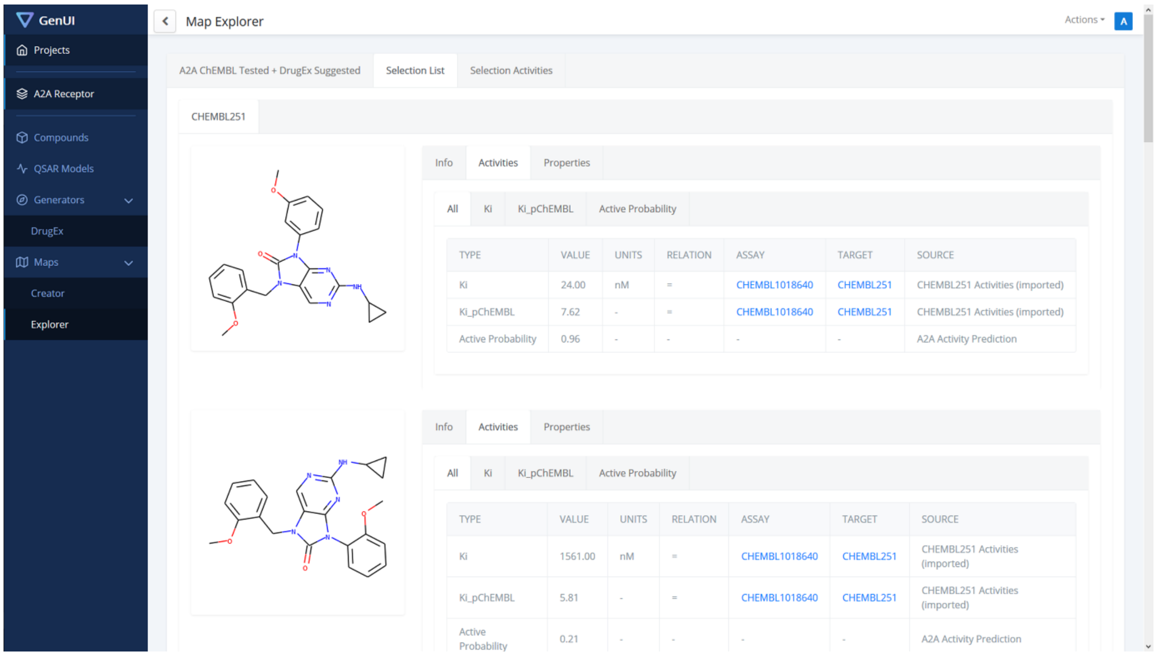
TYPE	VALUE	UNITS	SOURCE
Ki	24.00	nM	CHEMBL251 Activities (imported)
Ki_pChEMBL	7.62	-	CHEMBL251 Activities (imported)
Active Probability	0.96	-	A2A Activity Prediction

Fig. 6.8: A screenshot showing the "Explorer" part of the "Maps" GUI. The interactive plot on the left side of the screen is provided by the ChemSpace.js library [76]. Each point in this visualization corresponds to one molecule. In this particular configuration, the shapes and colors of the points indicate the compound set to which the compounds belong to. The color scheme of points can be changed with the menu in the top left corner of the plot. It is possible to color points by biological activities, physicochemical properties and other data associated with the compounds. The same can also be done with the size of the points. The points drawn in the map are interactive and hovering over a point shows a box with information about the compound inside and on the right side of the map.

6.3.6. Maps

Interactive visualization of chemical space is available under the “Maps” menu item. The menu separates the creation of the chemical space visualization, the “Creator” page (Fig. 6.7), and its exploration, the “Explorer” page (Fig. 6.8).

The “Creator” page is implemented as a grid of cards each of which represents an embedding of chemical compounds in 2D space (Fig. 6.8). Implicitly, the GenUI platform enables t-SNE [79] embedding (provided by openTSNE [80]). However, new projection methods can be easily added to the backend through the GenUI Python API with no need to modify the GUI (see Python API) [81].



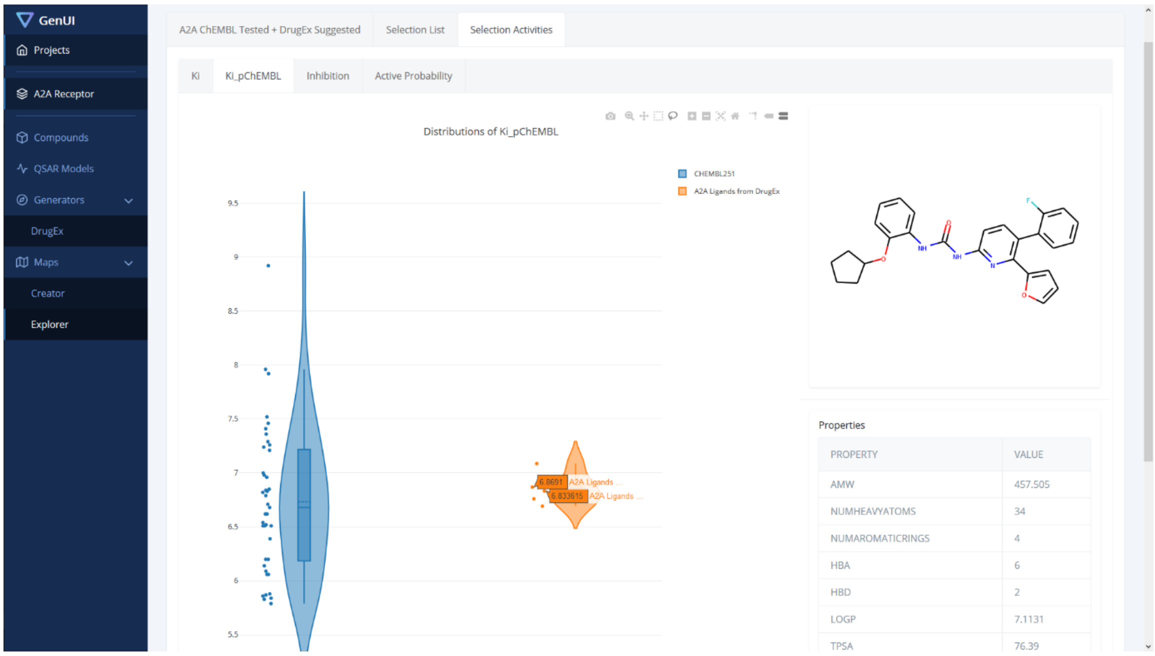
TYPE	VALUE	UNITS	RELATION	ASSAY	TARGET	SOURCE
Ki	24.00	nM	=	CHEMBL1018640	CHEMBL251	CHEMBL251 Activities (imported)
Ki_pChEMBL	7.62	-	=	CHEMBL1018640	CHEMBL251	CHEMBL251 Activities (imported)
Active Probability	0.96	-	-	-	-	A2A Activity Prediction

TYPE	VALUE	UNITS	RELATION	ASSAY	TARGET	SOURCE
Ki	1561.00	nM	=	CHEMBL1018640	CHEMBL251	CHEMBL251 Activities (imported)
Ki_pChEMBL	5.81	-	=	CHEMBL1018640	CHEMBL251	CHEMBL251 Activities (imported)
Active Probability	0.21	-	-	-	-	A2A Activity Prediction

Fig. 6.9: View of the “Selected List” tab of the “Explorer” page. The tab shows the selected molecules in the map as a list which is the same as the one used in the “Compounds” view (Fig. 6.4). For easier navigation, the compounds are also grouped by the compound set they belong to and the view for each set can be accessed by switching tabs above the displayed list (only one compound set, CHEMBL251, is present in this case).

The purpose of the “Explorer” page (Fig. 6.9) is to interactively visualize chemical space embedding prepared in the “Creator”. In the created visualization, users can explore compound bioactivities, physicochemical properties, and other measurements for various representations and parts of chemical space. Thanks to ChemSpace.js [76] up to 5 dimensions can be shown in the map at the same time with various visualization methods:

X and Y coordinates, point color, point size and point shape. The map can be zoomed in by drawing a rectangle over a group of points. Such points form a selection and their detailed information is displayed under the “Selected List” (Fig. 6.10) and “Selected Activities” tabs (Fig. 6.11).



Properties

PROPERTY	VALUE
AMW	457.505
NUMHEAVYATOMS	34
NUMAROMATICRINGS	4
HBA	6
HBD	2
LOGP	7.1131
TPSA	76.39

Fig. 6.10: View of the “Selection Activities” tab of the “Explorer” page. In this view, violin plots representing distributions of activities in the set of selected compounds are displayed. Each violin plot corresponds to one compound set and one activity type. The violin plots are also interactive and hovering over points updates the compound structure and its physicochemical properties are displayed on the right.

6.3.7. JavaScript API

Two main considerations in the development of GenUI are reusability and extensibility. Therefore, the frontend GUI comprises a large library of over 50 React components that are encapsulated in a standalone package (Fig. 6.12A). The package is organized into subpackages that follow the structure and hierarchy of design elements in the GenUI interface. In the following sections, we use the two most important groups of the React API components as case studies to illustrate how the frontend GUI can be extended. The presented components are “Model Components”, used to add new trainable models, and “REST API Components”, used to fetch and send data between the frontend and the GenUI REST API.

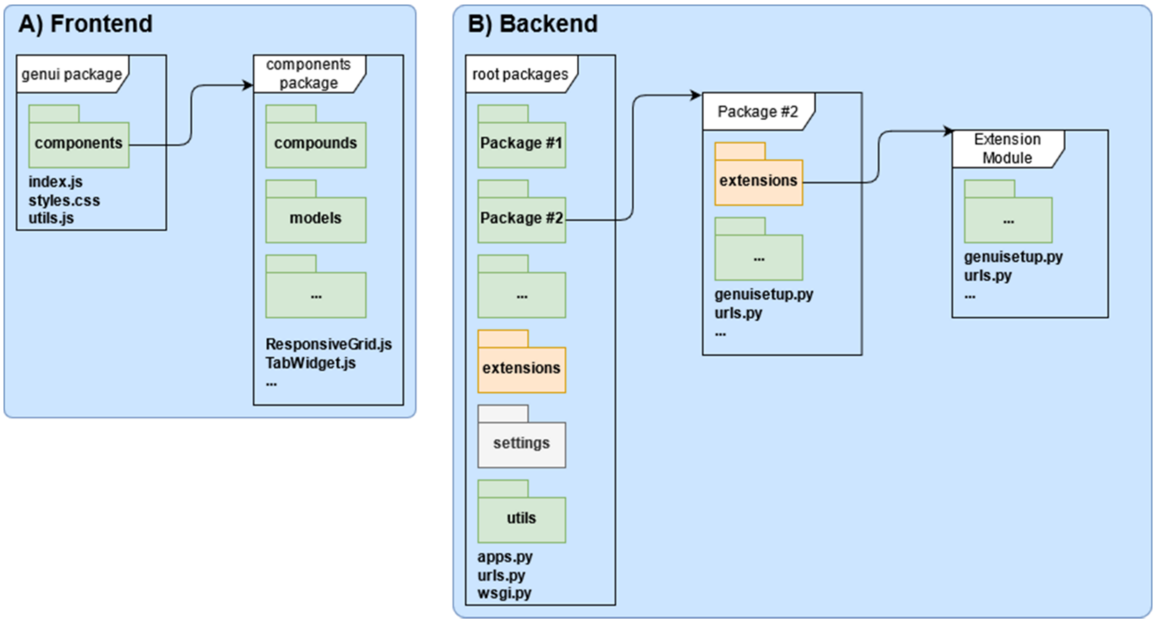


Fig. 6.11: Organization of the GenUI frontend (A) and backend (B) packages. The frontend React library (A) contains customized styles, utility functions and the React components used in the GenUI web client. The React components are further divided into groups related to the structure of the GenUI interface. Schematic depiction of the GenUI backend Python code. The backend (B) is structured as a standard Django project (designated by its settings package and the urls and wsgi modules). The GenUI code itself is divided into a number of root packages that are further divided into subpackages. The extensions subpackage is specific to GenUI and is used to automatically discover and configure extension modules. GenUI extensions and packages typically define the genuisetup module, which is used to configure the extension when the Django project is run.

6.3.8. Model components

Much of the functionality of the GenUI platform is based on trained models. The “QSAR Models”, “DrugEx” and “Maps” pages all borrow from the same library of reusable GenUI React components (Fig. 6.11A). At the core of the “models” component library (Fig. 6.11A) is the *ModelsPage* component (Fig. 6.12). *ModelsPage* manages the layout and data displayed in model cards. When the users select to build a new model, the *ModelsPage* component is also responsible to show a card with the model creation form. The information that the *ModelsPage* displays can be customized through various React properties (Fig. 6.12) that represent either data (data properties) or other components (component properties). Such an encapsulation approach and top-down data flow is one of the main strengths of the React framework. This design is very robust since it fosters appropriate separation of concerns by their encapsulation inside more and more specialized components. This makes the code easy to reuse and maintain.

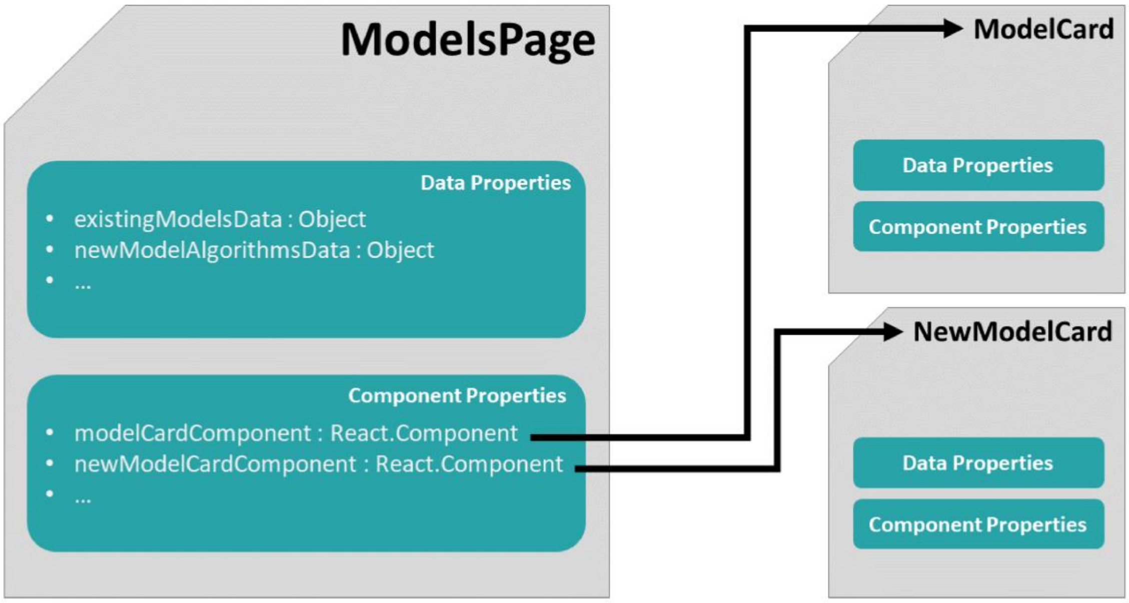


Fig. 6.12: A simplified illustration of the high-level components in the GenUI React API for rendering model cards. The main *ModelsPage* component has two kinds of attributes (called “properties” in React): (a) *data properties* and (b) *component properties*. The values of data properties are used to display model data while the values of component properties are used as child components and injected into the GUI at appropriate places. If no component property is specified, default components are used as children instead (i.e. *ModelCard* and *NewModelCard*). The child components can accept data and component properties as well from their parent (i.e *ModelsPage*). This creates a hierarchy of reusable components that can be easily assembled and configured to accommodate the different needs of each model view in a standardized and consistent manner.

6.3.9. REST API components

Because the GUI often needs to fetch data from the backend server, several React components were defined for that purpose. In order to use them, one just needs to provide the required REST API URLs as React component properties. For example, the *ComponentWithResources* component configured with the ‘/maps/algorithms/’ URL will get all available embedding methods as JSON (JavaScript Object Notation) and converts the result to a JavaScript object. Many components can also periodically update the fetched data, which is useful for tracking information in real time. For paginated data there is also the *ApiResourcePaginator* component that only fetches a new page if a given event is fired (i.e. user presses a button). This makes it convenient to create GUIs for larger data sets. In addition, user credentials are also handled automatically.

Many more specialized components are also available to fetch specific information. For example, the *TaskAwareComponent* tracks URLs associated with background asynchronous tasks and it regularly passes information about completed, running, or failed tasks to its child components. However, other specialized components exist that automatically fetch and format pictures of molecules, bioactivities, physicochemical properties or create, update and delete objects in the UI and the server [66].

6.4. Backend

The backend services are the core of the GenUI platform and the GenUI Python API provides a convenient way to write backend extensions (i.e. add new molecular generators, compound import filters, QSAR modeling algorithms, and dimensionality reduction methods for chemical space maps). All five backend services (Fig. 6.2) are implemented with the Django web framework [82] and Django REST Framework [83]. For data storage, a freely available Docker [70] image developed by Informatics Matters Ltd. [84] is used. The Docker image contains an instance of the PostgreSQL database system with integrated database cartridge from the RDKit cheminformatics framework [85]. The integration of RDKit with the Django web framework is handled with the Django RDKit library [86]. All compounds imported in the database are automatically standardized with the current version of the ChEMBL structure curation pipeline [87].

Because the backend services also handle processing of long-running and computationally intensive tasks, the framework uses Celery distributed task queue [88] with Redis as a message broker [89] to dispatch them to workers. Celery workers are processes running in the background that consume tasks from the task queue and process them asynchronously. Workers can either run on the same machine as the backend services or they can be distributed over an infrastructure of computers (see Deployment).

6.4.1. Python API

Django is a web framework that utilizes the Model View Template (MVT) design pattern to handle web requests and draw web pages. MVT is similar to the well-known Model View Controller (MVC) design pattern, but without a dedicated controller that determines what view needs to be called in response to a request. In MVT, the framework itself plays the role of the controller and makes sure that the correct view is called upon receiving a web request. In Django, the view is represented by a Python function or a method that returns various data types based on the nature of the request. The view can also take advantage of the Django templating engine to dynamically generate HTML pages. In both MVC and MVT, the model plays a role of a data access layer. The model represents the tables in the database and facilitates search and other data operations. GenUI does not use the Django templating engine, but rather handles all requests via REST API endpoints that manipulate data in JSON. This makes the frontend React application completely decoupled from the backend and also enables other clients to access the GenUI data in a convenient way by design (Fig. 6.1).

The GenUI backend codebase [67] follows the standard structure of any Django project and is divided into multiple Python packages that each encapsulate smaller self-contained parts (Fig. 6.12B). In GenUI, any package that resides in the root directory is referred to as the root package. Root packages facilitate many of the REST API endpoints (Fig. 6.2), but they also contain reusable classes that are intended to be built upon by extensions (see Generic Views and Viewsets, for example). In the following sections, some important features of the backend Python API are briefly highlighted. However, a much more detailed description with code examples is available on the documentation page of the project [81].

6.4.2. Extensions

Django is known for its strong focus on modularity and extensibility and GenUI tries to follow in its footsteps and support a flexible system of pluggable applications. Each of the GenUI root packages contains a Python package called *extensions* (Fig. 6.11B). The *extensions* package can contain any number of Django applications or Python modules, which ensures that the extending components of the GenUI framework are well-organized and loosely coupled.

Provided that GenUI extensions are structured a certain way they can take advantage of automatic configuration and integration (see Automatic Code Discovery). Before the Django project is deployed, GenUI applications and extensions are detected and configured with the *genuisetup* command, which makes sure that the associated REST API endpoints are exposed under the correct URLs. The *genuisetup* command is executed with the *manage.py* script (a utility script provided by the Django library).

6.4.3. Automatic code discovery

The root packages of the GenUI backend library define many abstract and generic base classes to implement and reuse in extensions. These classes either implement the REST API or define code to be run on the worker nodes inside Celery tasks. Automatic code discovery uses several introspection functions and methods to find the derived classes of the base classes found in the root packages. By default, this is done when the *genuisetup* command is executed (see Extensions).

For example, if the derived class defines a new machine learning algorithm to be used in QSAR modelling, automatic code discovery utilities make sure that the new algorithm appears as a choice in the QSAR modelling REST API and that proper parameter values are collected via the endpoint to create the model. Moreover, all changes also get automatically propagated to the web-based GUI because it uses the REST API to obtain algorithm choices for the model creation form. Thus, no JavaScript code has to be written to integrate a new machine learning algorithm. These concepts are also used when adding molecular generators, dimensionality reduction methods, or molecular descriptors.

6.4.4. Generic views and viewsets

When developing REST API services with the Django REST Framework [83], a common practice is to use generic views and sets of views (called viewsets). In Django applications, views are functions or classes that handle incoming HTTP requests. Viewsets are classes defined by the Django REST Framework that bring functionality of several views (such as creation, update or deletion of objects) into one single class. Generic views and viewsets are classes that usually do not stand on their own, but are designed to be further extended and customized.

The GenUI Python library embraces this philosophy and many REST API endpoints are encapsulated in generic views or viewsets. This ensures that the functionality can be reused and that no code needs to be written twice, as stated by the well-known DRY (“Don’t Repeat Yourself”) principle [90]. An example of such a generic approach is the *ModelViewSet* class that handles the endpoints for retrieval and training of machine learning models. This viewset is used by the *qsar* and *maps* applications, but also by the DrugEx extension. All these applications depend on some form of a machine learning model so they can take advantage of this interface, which automatically checks the validity of user inputs and sends model training jobs to the task queue.

6.4.5. Asynchronous tasks

Many of the GenUI backend services take advantage of asynchronous tasks which are functions executed in the background without blocking the main application. Moreover, tasks do not even have to be executed on the same machine as the caller of the task, which allows for a great deal of flexibility and scalability (see Deployment).

The Celery task queue [88] makes creating asynchronous tasks as easy as defining a Python function [91]. In addition, some GenUI views already define their own tasks and no explicit task definition is needed in the derived views of the extensions. For example, the *compounds* root package defines a generic viewset that can be used to create and manage compound sets. The import and creation of compounds belonging to a new compound set is handled by implementing a separate initializer class, which is passed to the appropriate generic viewset class [81]. The initialization of a compound set can take a long time or may fail and, thus, should be executed asynchronously. Therefore, the viewset of the *compounds* application automatically executes the methods of the initializer class asynchronously with the help of an available Celery worker.

6.4.6. Integration of new features with the two APIs

While a few examples of integrating new features to the GenUI platform have already been given for both Python and JavaScript, in this section a brief overview of all extensible features of the GenUI platform will be given. The vast majority of the features implemented in the reference platform presented in this work is realized through the extension system introduced earlier (see Extension). Extensions can use a wide selection of cheminformatics and data analysis tools each with their own level of complexity. Therefore, in this section we discuss the ease/difficulty of implementing the most common extensions and outline the problems the developers will face when developing each type of extension on both frontend and backend. All of the extensible use cases discussed here are also described in the project documentation with code examples [81].

6.4.7. Compounds import

Importing sets of compounds from various sources may require different approaches and as a result different kinds of interfaces. Therefore, the GenUI platform was designed with more flexibility in mind in this case. However, it also means that more configuration is needed from the developer. Extending the GenUI backend is accomplished by creating an extension application that defines the REST API URLs of the extension as well as views that will serve the defined URLs. GenUI provides a generic viewset class that can be derived from to make this process a matter of a few lines of code. The initializer class that handles the import itself also needs to be implemented by the developer of the extension, but an already prepared initializer base class is available in GenUI as well. Among other things, this base class also handles molecule standardization and clean up which ensures unified representation of chemical structures across data sets. In the frontend API, there is a selection of React components that can be used to build cards representing imported compound sets. The cards need very little configuration and automatically include metadata and the list of compounds in the compound set.

6.4.8. QSAR models

The backend model integration API is designed to provide easy and fast integration of simple machine learning algorithms even without the need to manually modify the frontend GUI. Adding a QSAR model can be as simple as adding a single class to the extension. The responsibility of this class is to use a machine learning algorithm to train and serialize a model upon receiving training data and predict unknown data from the deserialized model when requested. This class is also annotated with metadata about the model to be displayed in the frontend GUI. Therefore, in the simplest cases no URLs or customized GUI components need to be defined. The GenUI framework itself also performs cross-validation and independent set validation and data preprocessing. However, in many cases customized behavior, novel descriptor or validation metrics implementations might be necessary and in that case the developer may be required to define new URLs, views and modeling strategies. However, also in this case the GenUI platforms attempts to make this process easier by providing generic viewsets and loosely coupled base class implementations that the developers can take advantage of. In addition, the interface to define molecular descriptors and validation metrics is designed with reusability in mind and also exposes the implemented features to other QSAR algorithms if needed.

6.4.9. Molecular generators

Molecular generators can be of various types and even those based purely on DNNs are often of different architectures and take advantage of diverse software frameworks. GenUI is designed in a fashion that is agnostic to the type of algorithm used and it leaves preprocessing of the training data (if any) and the generation of output solely on the developer of the extension. GenUI only defines the means to communicate data between the framework and the generator code. This also means that integration of a molecular generator requires more customization the extent of which largely depends on the type of the generative algorithm used. The GenUI model integration API that is used for integration of QSAR models can also be used for integration of molecular generators based on DNNs and is used by the DrugEx extension. Therefore, integrating contemporary approaches that are mostly based on DNNs should be easier thanks to the possibility to follow the example of DrugEx as a proof of concept. Generators may also have different requirements on the information displayed in the GUI and, thus, it is expected that the GUI will be customized as well. However, if the generator takes advantage of the GenUI model integration API, this process is significantly simplified.

6.4.10. Chemical space maps

The dimensionality reduction methods used to create the chemical space maps shown in the GenUI interface are handled through the GenUI model integration API as well. Therefore, integration of these approaches is handled similarly to QSAR models and, thus, it comes with the same set of requirements and assumptions. Implementing a simple dimensionality reduction method will likely not require any steps beyond the definition of the one class that contains the implementing code and algorithm metadata.

6.5. Deployment

Since the GenUI platform consists of several components with many dependencies and spans multiple programming languages, it can be tedious to set up the whole project on a new system. Docker makes deployment of larger projects like this easier by encapsulating different parts of the deployment environment inside Docker images [68-70]. Docker images are simply downloaded and deployed on the target system without the need to install any other tools beside Docker. GenUI uses many official Docker images available on the Docker image sharing platform Docker Hub [92]. The PostgreSQL database with built-in RDKit cartridge [84], Redis [93] and the NGINX web server [94, 95] are all obtained by this standard channel. In addition, we defined the following images to support the deployment of the GenUI platform itself [71]:

1. *genui-main*: Used to deploy both the frontend web application and the backend services.
2. *genui-worker*: Deploys a basic Celery worker without GPU support.
3. *genui-gpuworker*: Deploys a Celery worker with GPU support. It is the same as the *genui-worker*, but it has the NVIDIA CUDA Toolkit already installed.

The tools to build these images are freely available [70]. Therefore, developers can create images for extended versions of the GenUI that fit the needs of their organizations. In addition, the separation of the main application (*genui-main*) from workers also allows distributed deployment over multiple machines, which opens up the possibility to create a scalable architecture that can quickly accommodate teams of varying sizes.

6.6. Future directions

Although the GenUI framework already implements much of the functionality needed to successfully integrate most molecular generators, there are still many aspects of the framework that can be improved. For instance, it would be beneficial if more sources of molecular structures and bioactivity information are integrated in the platform besides ChEMBL (i.e PubChem [95], ZINC [96], DrugBank [97], BindingDB [98] or Probes and Drugs [99]). Currently, GenUI also lacks features to perform effective similarity and substructure searches, which we see as a crucial next step to improve the appeal of the platform to medicinal chemists. The current version of GenUI would also benefit from extending the sets of descriptors, QSAR machine learning algorithms and chemical space projections since the performance of different methods can vary across data sets. Finally, the question of synthesizability of the generated structures should also be addressed and a system for predicting chemical reactions and retrosynthetic pathways could also be very useful to medicinal chemists if integrated in the GUI (i.e. by facilitating connection to a service such as the IBM RXN [100] or PostEra Manifold [101]).

Even though it is hard to determine the requirements of every project where molecular generators might be applied, many of the aforementioned features and improvements can be readily implemented with the GenUI React components (see JavaScript API) and the Python API (see Pyton API). In fact, the already presented extensions and the DrugEx interface are useful case studies that can be used as templates for integration of many other cheminformatics tools and *de novo* molecular generators. Therefore, we see GenUI as a flexible and scalable framework that can be used by organizations to quickly integrate tools and data the way it suits their needs the most. However, we would also like GenUI to become a new useful way to share the progress in the development of novel *de novo* drug design methods and other cheminformatics approaches in the public domain.

6.7. Conclusions

We implemented a full stack solution for integration of *de novo* molecular generation techniques in a multidisciplinary work environment. The proposed GenUI software platform provides a GUI designed to be easily understood by experts outside the cheminformatics domain, but it also offers a feature-rich REST API for programmatic access and straightforward integration with automated processes. The presented solution also provides extensive Python and JavaScript extension APIs for easy integration of new molecular generators and other cheminformatics tools. We envision that the field of molecular generation will likely expand in the future and that an open source software platform such as this one is a crucial step towards more widespread adoption of novel algorithms in drug discovery and related research. We also believe that GenUI can facilitate more engagement between different groups of users and inspire new directions in the field of *de novo* drug design.

Declarations

Authors' Contributions

GvW suggested the original idea of developing a graphical user interface for a molecular generator and supervised the study along with DS. MŠ extended the original idea and developed all software presented in this work. XL is the author of DrugEx and helped with its integration as a proof of concept. MŠ and XL also prepared the manuscript, which all authors proofread and agreed on.

Acknowledgements

Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the program Projects of Large Research, Development and Innovations Infrastructures.

XL thanks Chinese Scholarship Council (CSC) for funding.

Competing Interests

The authors declare that they have no competing interests.

Funding

This work was supported from the Ministry of Education of the Czech Republic (LM2018130).

Availability of Data and Materials

The complete GenUI codebase and documentation is distributed under the MIT license and located in three repositories publicly accessible on GitHub:

- https://github.com/martin-sicho/genui (backend Python code)
- https://github.com/martin-sicho/genui-gui (frontend React application)
- https://github.com/martin-sicho/genui-docker (Docker files and deployment scripts)

A reference application that was described in this manuscript can be deployed with Docker images that were uploaded to Docker Hub: https://hub.docker.com/u/sichom. However, the images can also be built with the available Docker files and scripts (archived at https://doi.org/10.5281/zenodo.4813625). The reference web application uses the following versions of the GenUI software:

- 0.0.0-alpha.1 for the frontend React application (archived at https://doi.org/10.5281/zenodo.4813608)
- 0.0.0.alpha1 for the backend Python application (archived at https://doi.org/10.5281/zenodo.4813586)

Availability and Requirements

- Project Name: GenUI
- Project Home Page: https://github.com/martin-sicho/genui
- Operating system(s): Linux
- Programming language: Python, JavaScript
- Other requirements: Docker 20.10.7 or higher
- License: MIT License

References

1. Wang Y, Cheng T, Bryant SH (2017) PubChem BioAssay: A Decade’s Development toward Open High-Throughput Screening Data Sharing. SLAS DISCOVERY: Advancing the Science of Drug Discovery 22(6):655-666.
2. Tetko IV, Engkvist O, Koch U, Reymond J-L, Chen H (2016) BIGCHEM: Challenges and Opportunities for Big Data Analysis in Chemistry. Molecular Informatics 35(11-12):615-621.
3. Rifaioglu AS, Atas H, Martin MJ, Cetin-Atalay R, Atalay V, Doğan T (2019) Recent applications of deep learning and machine intelligence on in silico drug discovery: methods, tools and databases. Brief Bioinform 20(5):1878-1912.
4. Hoffmann T, Gastreich M (2019) The next level in chemical space navigation: going far beyond enumerable compound libraries. Drug Discov Today 24(5):1148-1156.
5. Tetko IV, Engkvist O, Chen H (2016) Does ‘Big Data’ exist in medicinal chemistry, and if so, how can it be harnessed? Future medicinal chemistry 8(15):1801-1806.
6. Davies M, Nowotka M, Papadatos G, Dedman N, Gaulton A, Atkinson F, Bellis L, Overington JP (2015) ChEMBL web services: streamlining access to drug discovery data and utilities. Nucleic Acids Research 43(W1):W612-W620.
7. Mendez D, Gaulton A, Bento AP, Chambers J, De Veij M, Félix E, Magariños María P, Mosquera Juan F, Mutowo P, Nowotka M *et al* (2019) ChEMBL: towards direct deposition of bioassay data. Nucleic Acids Research 47(D1):D930-D940.
8. Polishchuk PG, Madzhidov TI, Varnek A (2013) Estimation of the size of drug-like chemical space based on GDB-17 data. Journal of Computer-Aided Molecular Design 27(8):675-679.
9. Drew KLM, Baiman H, Khwaounjoo P, Yu B, Reynisson J (2012) Size estimation of chemical space: how big is it? Journal of Pharmacy and Pharmacology 64(4):490-495.
10. Walters WP, Stahl MT, Murcko MA (1998) Virtual screening—an overview. Drug Discovery Today 3(4):160-178.
11. Bohacek RS, McMartin C, Guida WC (1996) The art and practice of structure-based drug design: A molecular modeling perspective. Med Res Rev 16(1):3-50.
12. Lenselink EB, ten Dijke N, Bongers B, Papadatos G, van Vlijmen HWT, Kowalczyk W, IJzerman AP, van Westen GJP (2017) Beyond the hype: deep neural networks outperform established methods using a ChEMBL bioactivity benchmark set. Journal of Cheminformatics 9(1):45.
13. Liu X, IJzerman AP, van Westen GJP. Computational Approaches for De Novo Drug Design: Past, Present, and Future. In: *Artificial Neural Networks.* Edited by Cartwright H. New York, NY: Springer US; 2021: 139-165.
14. Coley CW (2021) Defining and Exploring Chemical Spaces. Trends in Chemistry 3(2):133-145.
15. Opassi G, Gesù A, Massarotti A (2018) The hitchhiker’s guide to the chemical-biological galaxy. Drug Discovery Today 23(3):565-574.
16. Muratov EN, Bajorath J, Sheridan RP, Tetko IV, Filimonov D, Poroikov V, Oprea TI, Baskin II, Varnek A, Roitberg A *et al* (2020) QSAR without borders. Chemical Society Reviews 49(11):3525-3564.
17. Wang L, Ding J, Pan L, Cao D, Jiang H, Ding X (2019) Artificial intelligence facilitates drug design in the big data era. Chemometrics Intellig Lab Syst 194:103850.
18. Schneider G, Clark DE (2019) Automated De Novo Drug Design: Are We Nearly There Yet? Angew Chem Int Ed Engl 58(32):10792-10803.
19. Zhu H (2020) Big Data and Artificial Intelligence Modeling for Drug Discovery. Annual Review of Pharmacology and Toxicology 60(1):573-589.

20. Le TC, Winkler DA (2015) A Bright Future for Evolutionary Methods in Drug Design. ChemMedChem 10(8):1296-1300.
21. Lavecchia A (2019) Deep learning in drug discovery: opportunities, challenges and future prospects. Drug Discov Today 24(10):2017-2032.
22. Schreiber SL, Kotz JD, Li M, Aubé J, Austin CP, Reed JC, Rosen H, White EL, Sklar LA, Lindsley CW *et al* (2015) Advancing Biological Understanding and Therapeutics Discovery with Small-Molecule Probes. Cell 161(6):1252-1265.
23. Bian Y, Xie X-Q (2021) Generative chemistry: drug discovery with deep learning generative models. Journal of Molecular Modeling 27(3):71.
24. Zheng S, Lei Z, Ai H, Chen H, Deng D, Yang Y (2020) Deep Scaffold Hopping with Multi-modal Transformer Neural Networks.
25. Stojanović L, Popović M, Tijanić N, Rakočević G, Kalinić M (2020) Improved Scaffold Hopping in Ligand-Based Virtual Screening Using Neural Representation Learning. Journal of Chemical Information and Modeling 60(10):4629-4639.
26. Baskin II (2020) The power of deep learning to ligand-based novel drug discovery. Expert Opin Drug Discov 15(7):755-764.
27. Elton DC, Boukouvalas Z, Fuge MD, Chung PW (2019) Deep learning for molecular design—a review of the state of the art. Mol Syst Des Eng 4(4):828-849.
28. Xu Y, Lin K, Wang S, Wang L, Cai C, Song C, Lai L, Pei J (2019) Deep learning for molecular generation. Future Med Chem 11(6):567-597.
29. Jørgensen PB, Schmidt MN, Winther O (2018) Deep Generative Models for Molecular Science. Mol Inform 37(1-2).
30. Gantzer P, Creton B, Nieto-Draghi C (2020) Inverse-QSPR for de novo Design: A Review. Mol Inform 39(4):e1900087.
31. Yoshikawa N, Terayama K, Sumita M, Homma T, Oono K, Tsuda K (2018) Population-based De Novo Molecule Generation, Using Grammatical Evolution. Chem Lett 47(11):1431-1434.
32. Jensen JH (2019) A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. Chem Sci 10(12):3567-3572.
33. Spiegel JO, Durrant JD (2020) AutoGrow4: an open-source genetic algorithm for de novo drug design and lead optimization. J Cheminform 12(1):25.
34. Leguy J, Cauchy T, Glavatskikh M, Duval B, Da Mota B (2020) EvoMol: a flexible and interpretable evolutionary algorithm for unbiased de novo molecular generation. J Cheminform 12(1):55.
35. Hoksza D, Skoda P, Voršilák M, Svozil D (2014) Molpher: a software framework for systematic chemical space exploration. J Cheminform 6(1):7.
36. Schneider G, Fechner U (2005) Computer-based de novo design of drug-like molecules. Nature Reviews Drug Discovery 4(8):649-663.
37. Li X, Xu Y, Yao H, Lin K (2020) Chemical space exploration based on recurrent neural networks: applications in discovering kinase inhibitors. J Cheminform 12(1):42.
38. Grisoni F, Neuhaus CS, Hishinuma M, Gabernet G, Hiss JA, Kotera M, Schneider G (2019) De novo design of anticancer peptides by ensemble artificial neural networks. J Mol Model 25(5):112.
39. Wu J, Ma Y, Zhou H, Zhou L, Du S, Sun Y, Li W, Dong W, Wang R (2020) Identification of protein tyrosine phosphatase 1B (PTP1B) inhibitors through De Novo Evoluton, synthesis, biological evaluation and molecular dynamics simulation. Biochem Biophys Res Commun 526(1):273-280.
40. Polykovskiy D, Zhebrak A, Vetrov D, Ivanenkov Y, Aladinskiy V, Mamoshina P, Bozdaganyan M, Aliper A, Zhavoronkov A, Kadurin A (2018) Entangled Conditional Adversarial Autoencoder for de Novo Drug Discovery. Molecular Pharmaceutics 15(10):4398-4405.

41. Merk D, Friedrich L, Grisoni F, Schneider G (2018) De Novo Design of Bioactive Small Molecules by Artificial Intelligence. Molecular Informatics 37(1-2):1700153.

42. Putin E, Asadulaev A, Vanhaelen Q, Ivanenkov Y, Aladinskaya AV, Aliper A, Zhavoronkov A (2018) Adversarial Threshold Neural Computer for Molecular de Novo Design. Molecular Pharmaceutics 15(10):4386-4397.

43. Sumita M, Yang X, Ishihara S, Tamura R, Tsuda K (2018) Hunting for Organic Molecules with Artificial Intelligence: Molecules Optimized for Desired Excitation Energies. ACS Central Science 4(9):1126-1133.

44. Zhavoronkov A, Ivanenkov YA, Aliper A, Veselov MS, Aladinskiy VA, Aladinskaya AV, Terentiev VA, Polykovskiy DA, Kuznetsov MD, Asadulaev A *et al* (2019) Deep learning enables rapid identification of potent DDR1 kinase inhibitors. Nature Biotechnology 37(9):1038-1040.

45. Grisoni F, Huisman BJH, Button AL, Moret M, Atz K, Merk D, Schneider G (2021) Combining generative artificial intelligence and on-chip synthesis for de novo drug design. Science Advances 7(24):eabg3338.

46. Sparkes A, Aubrey W, Byrne E, Clare A, Khan MN, Liakata M, Markham M, Rowland J, Soldatova LN, Whelan KE *et al* (2010) Towards Robot Scientists for autonomous scientific discovery. Autom Exp 2:1.

47. Coley CW, Eyke NS, Jensen KF (2020) Autonomous Discovery in the Chemical Sciences Part I: Progress. Angewandte Chemie International Edition 59(51):22858-22893.

48. Coley CW, Eyke NS, Jensen KF (2020) Autonomous Discovery in the Chemical Sciences Part II: Outlook. Angewandte Chemie International Edition 59(52):23414-23436.

49. Henson AB, Gromski PS, Cronin L (2018) Designing Algorithms To Aid Discovery by Chemical Robots. ACS Cent Sci 4(7):793-804.

50. Dimitrov T, Kreisbeck C, Becker JS, Aspuru-Guzik A, Saikin SK (2019) Autonomous Molecular Design: Then and Now. ACS Appl Mater Interfaces 11(28):24825-24836.

51. Schneider G (2018) Automating drug discovery. Nat Rev Drug Discov 17(2):97-113.

52. Accelerys I (2013) Accelerys Metabolite Database. In. San Diego.

53. Willems H, De Cesco S, Svensson F (2020) Computational Chemistry on a Budget: Supporting Drug Discovery with Limited Resources. J Med Chem 63(18):10158-10169.

54. Chu Y, He X (2019) MoleGear: A Java-Based Platform for Evolutionary De Novo Molecular Design. Molecules 24(7).

55. Douguet D (2010) e-LEA3D: a computational-aided drug design web server. Nucleic Acids Research 38(suppl_2):W615-W621.

56. Winter R, Retel J, Noé F, Clevert D-A, Steffen A (2020) grünifai: interactive multiparameter optimization of molecules in a continuous vector space. Bioinformatics 36(13):4093-4094.

57. Green H, Koes DR, Durrant JD (2021) DeepFrag: a deep convolutional neural network for fragment-based lead optimization. Chemical Science 12(23):8036-8047.

58. Sommer K, Flachsenberg F, Rarey M (2019) NAOMInext – Synthetically feasible fragment growing in a structure-based design context. European Journal of Medicinal Chemistry 163:747-762.

59. Pastor M, Gómez-Tamayo JC, Sanz F (2021) Flame: an open source framework for model development, hosting, and usage in production environments. Journal of Cheminformatics 13(1):31.

60. Green DVS, Pickett S, Luscombe C, Senger S, Marcus D, Meslamani J, Brett D, Powell A, Masson J (2020) BRADSHAW: a system for automated molecular design. Journal of Computer-Aided Molecular Design 34(7):747-765.

61. Ivanenkov YA, Zhebrak A, Bezrukov D, Zagribelnyy B, Aladinskiy V, Polykovskiy D, Putin E, Kamya P, Aliper A, Zhavoronkov A (2021) Chemistry42: An AI-based platform for de novo

molecular design. arXiv preprint arXiv:210109050.

62. Zhumagambetov R, Kazbek D, Shakipov M, Maksut D, Peshkov VA, Fazli S (2020) cheML.io: an online database of ML-generated molecules. RSC Advances 10(73):45189-45198.
63. Griffen EJ, Dossetter AG, Leach AG (2020) Chemists: AI Is Here; Unite To Get the Benefits. Journal of Medicinal Chemistry 63(16):8695-8704.
64. Liu X, Ye K, van Vlijmen HWT, IJzerman AP, van Westen GJP (2019) An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine A2A receptor. J Cheminform 11(1):35.
65. MIT License. https://opensource.org/licenses/MIT. Accessed 2021-03-12.
66. GenUI Frontend Application. By Šícho M. https://github.com/martin-sicho/genui-gui. Accessed 2021-03-12.
67. GenUI Backend Application. https://github.com/martin-sicho/genui Accessed 2020-05-03.
68. Merkel D (2014) Docker: lightweight Linux containers for consistent development and deployment. Linux J 2014(239):Article 2.
69. Cito J, Ferme V, Gall HC: Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research. In: *Web Engineering: 2016// 2016; Cham*. Springer International Publishing: 609-612.
70. Docker. https://github.com/docker/docker-ce. Accessed 2020-05-03.
71. GenUI Docker Files. By Šícho M. https://github.com/martin-sicho/genui-docker. Accessed 2020-05-03.
72. React: A JavaScript Library for Building User Interfaces. By Facebook I. https://reactjs.org/. Accessed 2020-12-16.
73. Vibe: A beautiful react.js dashboard build with Bootstrap 4. By Salas J. https://github.com/NiceDash/Vibe. Accessed 2020-05-03.
74. Tétreault-Pinard ÉO (2019) Plotly JavaScript Open Source Graphing Library.
75. Chart.js: Simple yet flexible JavaScript charting for designers & developers. https://www.chartjs.org/. Accessed 2020-05-03.
76. ChemSpace JS. https://openscreen.cz/software/chemspace/home/. Accessed 2020-05-03.
77. Schaduangrat N, Lampa S, Simeon S, Gleeson MP, Spjuth O, Nantasenamat C (2020) Towards reproducible computational drug discovery. J Cheminform 12(1):9.
78. Liu X, Ye K, van Vlijmen HWT, IJzerman AP, van Westen GJP (2019) An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine A2A receptor. Journal of Cheminformatics 11(1):35.
79. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. Journal of Machine Learning Research 9:2579-2605.
80. Poličar PG, Stražar M, Zupan B (2019) openTSNE: a modular Python library for t-SNE dimensionality reduction and embedding. bioRxiv:731877.
81. GenUI Python Documentation. https://martin-sicho.github.io/genui/docs/index.html. Accessed 2021-03-12.
82. Foundation DS (2019) Django (Version 2.2).
83. Encode OSS L (2019) Django REST Framework.
84. Debian-based images containing PostgreSQL with the RDKit cartridge. https://hub.docker.com/r/informaticsmatters/rdkit-cartridge-debian. Accessed 2020-05-03.
85. RDKit: Open-source cheminformatics toolkit. By http://www.rdkit.org/. Accessed 2020-05-03.
86. Django RDKit. https://github.com/rdkit/django-rdkit. Accessed 2020-05-03.
87. Bento AP, Hersey A, Félix E, Landrum G, Gaulton A, Atkinson F, Bellis LJ, De Veij M, Leach AR

(2020) An open source chemical structure curation pipeline using RDKit. J Cheminform 12(1):51.

88. CELERY: Distributed Task Queue. https://github.com/celery/celery. Accessed 2020-05-03.

89. Redis: in-memory data structure store. By https://github.com/redis/redis. Accessed 2020-05-03.

90. Hunt A, Thomas D (2000). The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley Longman Publishing Co. Inc.

91. Celery: Get Started. https://docs.celeryproject.org/en/stable/getting-started/introduction.html#get-started. Accessed 2020-12-16.

92. Docker Hub. https://hub.docker.com/. Accessed 2020-12-16.

93. Redis: Docker Official Images. By https://hub.docker.com/_/redis. Accessed 2020-05-03.

94. NGINX Web Server. By https://github.com/nginx/nginx. Accessed 2020-05-03.

95. NGINX: Official Docker Images. By https://hub.docker.com/_/nginx. Accessed 2020-05-03.

96. Kim S, Chen J, Cheng T, Gindulyte A, He J, He S, Li Q, Shoemaker BA, Thiessen PA, Yu B *et al* (2019) PubChem 2019 update: improved access to chemical data. Nucleic Acids Research 47(D1):D1102-D1109.

97. Irwin JJ, Sterling T, Mysinger MM, Bolstad ES, Coleman RG (2012) ZINC: A Free Tool to Discover Chemistry for Biology. Journal of Chemical Information and Modeling 52(7):1757-1768.

98. Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, Chang Z, Woolsey J (2006) DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Research 34(suppl_1):D668-D672.

99. Gilson MK, Liu T, Baitaluk M, Nicola G, Hwang L, Chong J (2016) BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology. Nucleic Acids Research 44(D1):D1045-D1053.

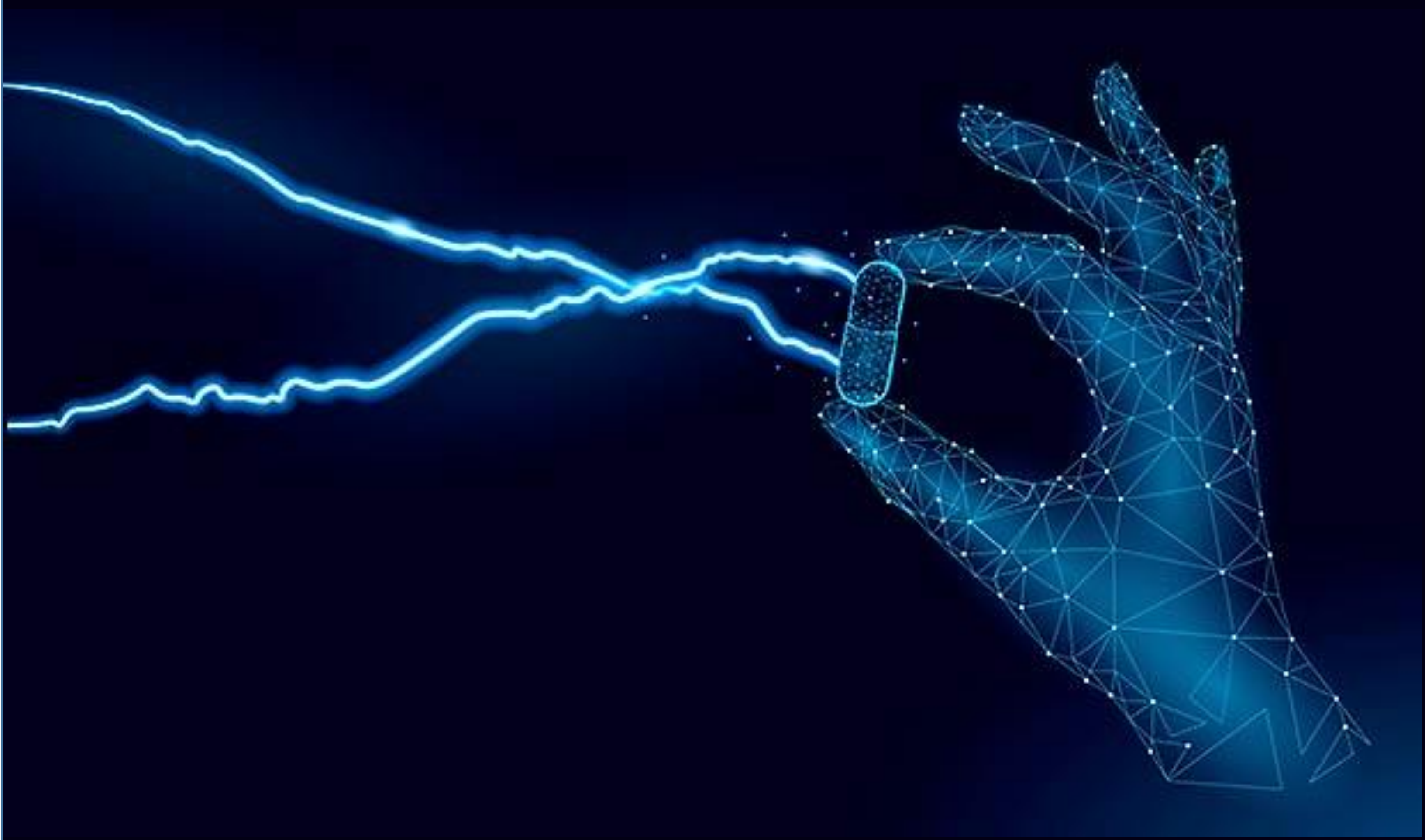
100. Skuta C, Popr M, Muller T, Jindrich J, Kahle M, Sedlak D, Svozil D, Bartunek P (2017) Probes & Drugs portal: an interactive, open data resource for chemical biology. Nature Methods 14(8):759-760.

101. IBM RXN for Chemistry. https://rxn.res.ibm.com/. Accessed 2021-03-12.

102. PostEra Manifold. https://postera.ai/manifold/. Accessed 2021-03-12.

Chapter 7

Conclusions and future perspectives



Having provided a review about computational approaches for *de novo* drug design and four research projects in the previous chapters, I am well versed in cutting-edge AI technologies, especially deep learning, applied in different scenarios of *de novo* drug design. In the following paragraphs, I will draw conclusions of this thesis and give a future outlook to illustrate its appropriateness in drug discovery and to bring forward other promising scopes for its application.

7.1. Conclusions

Drug discovery is a time- and resource-consuming process. To this end, computational approaches that are applied in *de novo* drug design play an important role to improve the efficiency and decrease the costs to develop novel drugs. Over several decades, a variety of methods have been proposed and applied in practice [1]. Traditionally, drug design problems are always taken as the combinational optimization in discrete chemical space, such as evolutionary algorithms [2,3], heuristic search algorithms [4], simulated annealing algorithms [5], *etc.*. Hence optimization methods were exploited to search for new drug molecules that meet multiple objectives. With the accumulation of data and the development of machine learning methods, computational drug design methods have gradually shifted to a new paradigm. There has been particular interest in the potential application of deep learning methods to drug design [6]. In **Chapter 2**, we gave a brief description of these two different *de novo* methods, compared their application scopes and discussed their possible development in the future.

Over the last ten years deep learning has progressed tremendously in both image recognition, natural language processing and other data rich fields [7]. In drug discovery, recurrent neural networks (RNNs) have been shown to be an effective method to generate novel chemical structures in the form of SMILES [8]. However, ligands generated by current methods have so far provided relatively low diversity and do not fully cover the whole chemical space occupied by known ligands. In **Chapter 3**, we therefore propose a new method (*DrugEx*) to discover *de novo* drug-like molecules. *DrugEx* is an RNN model (generator) trained through a special exploration strategy integrated into reinforcement learning. As a case study we applied our method to design ligands for the adenosine A_{2A} receptor. From ChEMBL data, a machine learning model (predictor) was created to predict whether generated molecules are active or not. Based on this predictor as the reward function, the generator was trained by reinforcement learning without any further data. We then compared the performance of our method with two previously published methods, *REINVENT* [9] and *ORGANIC* [10]. We found that the candidate molecules our model designed and predicted to be active, had a larger chemical diversity and better covered the chemical space of known ligands compared to the state-of-the-art (SOTA).

Although deep learning has led to breakthroughs in drug discovery, most of its applications only focus on a single drug target to generate drug-like active molecules. This is in spite of the reality that drug molecules often interact with more than one target which can have desired (polypharmacology) or undesired (toxicity) effects. In polypharmacology ideal drugs are required to bind to multiple specific targets to enhance efficacy or to reduce the development of resistance [11]. In **Chapter 4**, we extended our *DrugEx* algorithm with multi-objective optimization to generate drug molecules towards multiple targets or one specific target while avoiding off-targets (the two adenosine receptors, A_1AR and A_{2A}AR, and the potassium ion channel hERG). In our model, we applied an RNN as the *agent* and machine learning predictors as the *environment*, both of which were pre-trained in advance and then interplayed under the reinforcement learning framework. The concept of evolutionary algorithms was merged into our method such that *crossover* and *mutation* operations were implemented by the same deep learning model as the *agent*. During the training loop, the agent generates a batch of SMILES-based molecules. Subsequently scores for all objectives provided by the *environment* are used for constructing Pareto ranks of the generated molecules with non-dominated sorting and Tanimoto-based crowding distance algorithms. Here, we adopted GPU acceleration to speed up the process of Pareto optimization. The final reward of each molecule is calculated based on the Pareto ranking with the ranking selection algorithm [12]. The agent is trained under the guidance of the reward to make sure it can generate more desired molecules after convergence of the training process. All in all we demonstrated the generation of compounds with a diverse predicted selectivity profile toward multiple targets, offering the potential of high efficacy and lower toxicity.

Due to the huge chemical space in which feasible drug-like molecules are searched for, rational drug design always starts from specific molecular scaffolds as the core to which side chains are added or modified. With the rapid growth of deep learning methods and their application in drug discovery, a variety of approaches has been developed for *de novo* drug design. However, earlier versions of *DrugEx* are trained under fixed objectives and do not allow users to input any prior information, like most goal-directed methods. In order to improve its generality, *DrugEx* was updated to design drug molecules based on multiple scaffolds given by users. In **Chapter 5** we extended the transformer model [13], which is a multi-head self-attention deep learning model containing an encoder and a decoder, to deal with each molecule as a graph. The encoder of the graph transformer receives the input graph of the scaffolds containing multiple fragments and its decoder outputs the graph-based molecule containing given scaffolds. Each molecule was generated by growing and connecting procedures for the fragments in given scaffolds that were unified into one model. Moreover, we trained this generator under the reinforcement learning framework to increase the number of active ligands. As proof our proposed method was applied to design adenosine A_{2A} receptor ligands which were compared with SMILES-based methods. The results demonstrated its effectiveness as most of the generated molecules contained the given scaffolds and had a high virtual affinity towards the adenosine A_{2A} receptor.

Despite the rapid growth of AI techniques in drug discovery, widespread adoption of new *de novo* drug design approaches in the fields of medicinal chemistry and chemical biology is still lagging behind the most recent developments. It is urgently needed to establish a close collaboration between diverse teams of experimental and theoretical scientists. To accelerate the adoption of both modern and traditional *de novo* molecular generators, we developed *GenUI* (Generator User Interface), a software platform that makes it possible to integrate molecular generators within a feature-rich graphical user interface that is easy to use by experts of varying backgrounds. *GenUI* is implemented as a web service and its interfaces offer access to cheminformatics tools for data preprocessing, model building, molecule generation, and interactive chemical space visualization. Moreover, the platform is easy to extend with customizable frontend React.js components and backend Python extensions. *GenUI* is open source which has integrated *DrugEx* as a proof of principle. In **Chapter 6**, we presented the architecture and implementation details of GenUI and discuss how it can facilitate collaboration in the disparate communities interested in *de novo* molecular generation and computer-aided drug discovery.

7.2. Further perspectives

With the four projects mentioned above we catch a glimpse of the overwhelming power of AI in drug *de novo* design. However, it is impossible to make a thorough investigation of its capability in every scope of drug discovery with only four years study. In my view, there are still a plethora of promising issues about the development and application of AI to design chemical compounds that attract researchers' interest and are worth addressing.

7.2.1. New AI technologies

Deep learning is the most attractive branch in AI and it is still growing rapidly. First convolutional neural networks and recurrent neural networks achieved a breakthrough in image recognition and natural language processing [7]. Consequently the transformer model was proposed based on a self-attention mechanism in 2017 and achieved SOTA performance in language processing [13]. Subsequently, a large number of variants have been developed. For example, BERT, which is the encoder part of the transformer and is pre-trained with large amounts of data, improved the performance of sequence data prediction dramatically [14]. This led to more and more researchers employing it to construct predictive models for biological and chemical data [15,16]. In addition, GPT-3, which is also derived from the transformer model, achieved SOTA performance in many sequence generation tasks [17]. Moreover, transformer-based methods can also deal with graph data [18], allowing it to be applied to graph-based molecular design. Therefore they are promising algorithms to be used in drug design.

With respect to the huge number of parameters in the complicated architectures of deep learning, there are also many new methods to effectively train these models. For example, when dealing with image generation, generative adversarial networks [10] and variational autoencoders [19] are commonly used to train the model to generate the most similar samples. When introducing different computational methods in drug design in **Chapter 2**, we discussed the possibility of the combination of deep learning methods and optimization methods. Afterwards, we proposed a new kind of training method through simulating the idea of evolutionary algorithms **in Chapter 4**. Moreover, there are many studies about the application of evolutionary algorithms [20] or Bayesian optimization [21] to update the parameters in deep learning models. With the architecture of deep learning becoming more and more complex, it is worth discussing about how to effectively train models to avoid the issues of the local minimum and overfitting.

7.2.2. Different constraint conditions

Besides the objectives mentioned in the previous chapters, such as affinity for adenosine receptors and the drug-likeness score, the ideal drug molecule also needs to meet more objectives in reality. In addition to the affinity for one or more given targets, it also needs to have qualified ADME (absorption, distribution, metabolism, and excretion) properties [22] and low toxicity. More specifically, some of these requirements can be conflicting and cannot be satisfied simultaneously. Therefore, an important issue is to orchestrate the many objectives for effective drug design. However, most of current studies just simply transform the multi-objectives into a single objective with the weighted sum of these scores in order to guide the training of deep learning models. Actually, there are plenty of multi-objective optimization methods [12] being developed as mentioned in **Chapter 2**. These methods are worth exploring their integration with deep learning models.

Another important property of generated drug molecules is their synthesizability. However, the most current SMILES-based and Graph-based models cannot directly guarantee that the generated molecules can be synthesized [23]. Therefore, it is critical to predict the synthesizability of these generated molecules, which determines if they could be experimentally tested in practice. For example, some researchers combined deep reinforcement learning and Monto Carlo tree search to put forward to methods to predict retrosynthesis score [24,25] for given molecules and provide the feasible synthetic schemes [26]. Moreover, some other groups directly generate molecule base on reaction, in which each molecule in the training set are decomposed as a reaction tree [27]. And the aim of the model is choosing the reaction from the library step by step. In the end, the molecule construct with the whole reaction tree is generated.

In **Chapter 3 & 4**, all of the model conditions were fixed. This allowed the model to be trained well, but it cannot interact with users by receiving continued and updated information. If the conditions are changed, the model has to be trained again, which is an inconvenient and time-consuming process. In order to improve the generality of the model we proposed a new method in **Chapter 5** in which an end-to-end model received scaffold information from users. General speaking, it can also take other information as input to design bioactive molecules conditionally. For example, it can be used for lead optimization, *i.e.* the input can be an inactive or toxic ligand, and the output should be a similar ligand but active or safe, respectively. Moreover, now that proteochemometric modelling (PCM) has been proposed for many years to take the information of both drug and target information as input and predict their affinity [28], it can also be used to construct inverse PCM models, which take protein information as input to design its active ligands [29]. Considering that the full sequence length of some proteins is too large to be dealt with by current deep learning models, protein descriptors can also be used as input information.

7.2.3. Designing various kind of molecules

In this thesis, we only focus on the generation of small organic molecules, but there are other biological/chemical molecules to be designed. For example, natural products have always been the effective components of traditional Chinese medicine, but their physico-chemical properties are distinct from classical drug molecules. For instance unlike small synthetic molecules most of the natural products do not adhere to the Rule of 5 [30]. Compared with classical drug molecules, natural products also have different advantages as drug candidates. Natural products have been optimized by long-term natural evolution to have particular bioactivities, including the regulation of endogenous defense mechanisms through the interaction with other organisms, which is the possible reason for its key role in therapeutic areas especially for infectious diseases and cancer [31]. Moreover, their use in traditional medicine may provide insights regarding efficacy and safety, covering a wider area of chemical space compared with small organic molecules [32]. Now that there are several AI methods for the retrosynthesis of organic molecules [26], they also provide a valuable direction to exploit these methods in the synthesis pathway prediction of natural products.

Besides small organic molecules, peptides and proteins are important macromolecules for medicine. For example, some antimicrobial peptides can be used as drugs to inhibit the growth of a variety of microbes. The data representation of a peptide is a sequence of amino acid residues, which is feasible to be designed with deep learning models [33]. Moreover, there are variable domains in Fab regions of antibodies which determine specificity and efficacy to recognize the antigen. This part of the antibody also needs to be designed and can be generated by AI methods [34].

7.3. Final notes

The main thrust of this thesis is a comprehensive study about the application of AI technologies in *de novo* drug design. An integrative Python-based toolkit named *DrugEx* was developed to facilitate the accessibility of our methods to other researchers. In order to decrease the threshold for experimental researchers who are not familiar with computer coding, this tool was also used as the engine integrated into a web-based graphic toolkit named *GenUI* which has powerful capabilities of interactions with users and developers. These two software packages are my main contributions to the scientific community. Generally speaking, the highlight of this thesis is sufficiently embodied on the cover page. Faced with the huge chemical space of drug-like molecules (unveiling of the capsule at the bottom), AI is an effective approach to rapidly narrow down the search scope. AI itself is a mimic of the human brain running *in silico* (the logo in the center). The chip located in the center of the brain consists of a variety of different electronic components. Seven tandem diodes resemble the protein structure of a GPCR which has seven transmembrane domains. Its intracellular domain with the G protein (a total of four subunits represented by four gears) forms a virtual document which recodes with digits if the GPCR is activated or not. The component in the lower right side is like a magnifying glass that is identifying the active ligands after exploring the huge chemical space with this virtual lab. I hope the readers could be beneficial from this thesis to have broad and deep understanding of the role that AI methods play in drug discovery.

Reference

1. Schneider G, Fechner U (2005) Computer-based de novo design of drug-like molecules. Nat Rev Drug Discov 4 (8):649-663. doi:10.1038/nrd1799
2. van der Horst E, Marques-Gallego P, Mulder-Krieger T, van Veldhoven J, Kruisselbrink J, Aleman A, Emmerich MT, Brussee J, Bender A, Ijzerman AP (2012) Multi-objective evolutionary design of adenosine receptor ligands. Journal of chemical information and modeling 52 (7):1713-1721. doi:10.1021/ci2005115
3. Lameijer EW, Kok JN, Back T, Ijzerman AP (2006) The molecule evoluator. An interactive evolutionary algorithm for the design of drug-like molecules. Journal of chemical information and modeling 46 (2):545-552. doi:10.1021/ci050369d
4. Gillet VJ, Newell W, Mata P, Myatt G, Sike S, Zsoldos Z, Johnson AP (1994) SPROUT: recent developments in the de novo design of molecules. J Chem Inf Comput Sci 34 (1):207-217. doi:10.1021/ci00017a027
5. Sengupta S, Bandyopadhyay S (2012) De novo design of potential RecA inhibitors using multi objective optimization. IEEE/ACM Trans Comput Biol Bioinform 9 (4):1139-1154. doi:10.1109/TCBB.2012.35
6. Chen H, Engkvist O, Wang Y, Olivecrona M, Blaschke T (2018) The rise of deep learning in drug discovery. Drug discovery today. doi:10.1016/j.drudis.2018.01.039
7. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521 (7553):436-444. doi:10.1038/nature14539
8. Segler MHS, Kogej T, Tyrchan C, Waller MP (2018) Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. ACS Cent Sci 4 (1):120-131. doi:10.1021/acscentsci.7b00512
9. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. Journal of cheminformatics 9 (1):48. doi:10.1186/s13321-017-0235-x
10. Benjamin S-L, Carlos O, Gabriel L. G, Alan A-G (2017) Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC). doi:10.26434/chemrxiv.5309668.v3
11. Chaudhari R, Tan Z, Huang B, Zhang S (2017) Computational polypharmacology: a new paradigm for drug discovery. Expert Opin Drug Discov 12 (3):279-291. doi:10.1080/17460441.2017.1280024
12. Emmerich MTM, Deutz AH (2018) A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Nat Comput 17 (3):585-609. doi:10.1007/s11047-018-9685-y
13. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin IJae-p (2017) Attention Is All You Need.arXiv:1706.03762
14. Devlin J, Chang M-W, Lee K, Toutanova KJae-p (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.arXiv:1810.04805
15. Chithrananda S, Grand G, Ramsundar BJae-p (2020) ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction.arXiv:2010.09885
16. Wang S, Guo Y, Wang Y, Sun H, Huang J (2019) SMILES-BERT: Large Scale Unsupervised Pre-Training for Molecular Property Prediction. Paper presented at the Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Niagara Falls, NY, USA,
17. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C,

McCandlish S, Radford A, Sutskever I, Amodei DJae-p (2020) Language Models are Few-Shot Learners.arXiv:2005.14165

18. Yun S, Jeong M, Kim R, Kang J, Kim HJJae-p (2019) Graph Transformer Networks.arXiv:1911.06455
19. Gomez-Bombarelli R, Wei JN, Duvenaud D, Hernandez-Lobato JM, Sanchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel TD, Adams RP, Aspuru-Guzik A (2018) Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. ACS Cent Sci 4 (2):268-276. doi:10.1021/acscentsci.7b00572
20. Young SR, Rose DC, Karnowski TP, Lim S-H, Patton RM (2015) Optimizing deep learning hyper-parameters through an evolutionary algorithm. Paper presented at the Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, Austin, Texas,
21. Griffiths RR, Hernandez-Lobato JM (2020) Constrained Bayesian optimization for automatic chemical design using variational autoencoders. Chem Sci 11 (2):577-586. doi:10.1039/c9sc04026a
22. Kirchmair J, Goller AH, Lang D, Kunze J, Testa B, Wilson ID, Glen RC, Schneider G (2015) Predicting drug metabolism: experiment and/or computation? Nat Rev Drug Discov 14 (6):387-404. doi:10.1038/nrd4581
23. Liu X, IJzerman AP, van Westen GJP (2021) Computational Approaches for De Novo Drug Design: Past, Present, and Future. Methods Mol Biol 2190:139-165. doi:10.1007/978-1-0716-0826-5_6
24. Genheden S, Thakkar A, Chadimova V, Reymond JL, Engkvist O, Bjerrum E (2020) AiZynthFinder: a fast, robust and flexible open-source software for retrosynthetic planning. Journal of cheminformatics 12 (1):70. doi:10.1186/s13321-020-00472-1
25. Thakkar A, Chadimova V, Bjerrum EJ, Engkvist O, Reymond JL (2021) Retrosynthetic accessibility score (RAscore) - rapid machine learned synthesizability classification from AI driven retrosynthetic planning. Chem Sci 12 (9):3339-3349. doi:10.1039/d0sc05401a
26. Segler MHS, Preuss M, Waller MP (2018) Planning chemical syntheses with deep neural networks and symbolic AI. Nature 555 (7698):604-610. doi:10.1038/nature25978
27. Ghiandoni GM, Bodkin MJ, Chen B, Hristozov D, Wallace JEA, Webster J, Gillet VJ (2020) Enhancing reaction-based de novo design using a multi-label reaction class recommender. J Comput Aided Mol Des 34 (7):783-803. doi:10.1007/s10822-020-00300-6
28. van Westen GJ, Wegner JK, Geluykens P, Kwanten L, Vereycken I, Peeters A, Ijzerman AP, van Vlijmen HW, Bender A (2011) Which compound to select in lead optimization? Prospectively validated proteochemometric models guide preclinical development. PLoS One 6 (11):e27518. doi:10.1371/journal.pone.0027518
29. Grechishnikova D (2021) Transformer neural network for protein-specific de novo drug generation as a machine translation problem. Sci Rep 11 (1):321. doi:10.1038/s41598-020-79682-4
30. Mullard A (2018) Re-assessing the rule of 5, two decades on. Nat Rev Drug Discov 17 (11):777. doi:10.1038/nrd.2018.197
31. Atanasov AG, Waltenberger B, Pferschy-Wenzig EM, Linder T, Wawrosch C, Uhrin P, Temml V, Wang L, Schwaiger S, Heiss EH, Rollinger JM, Schuster D, Breuss JM, Bochkov V, Mihovilovic MD, Kopp B, Bauer R, Dirsch VM, Stuppner H (2015) Discovery and resupply of pharmacologically active plant-derived natural products: A review. Biotechnol Adv 33 (8):1582-1614. doi:10.1016/j.biotechadv.2015.08.001
32. Atanasov AG, Zotchev SB, Dirsch VM, International Natural Product Sciences T, Supuran CT (2021) Natural products in drug discovery: advances and opportunities. Nat Rev Drug Discov 20 (3):200-216. doi:10.1038/s41573-020-00114-z
33. Wang C, Garlick S, Zloh M (2021) Deep Learning for Novel Antimicrobial Peptide Design.

Biomolecules 11 (3). doi:10.3390/biom11030471

34. Saka K, Kakuzaki T, Metsugi S, Kashiwagi D, Yoshida K, Wada M, Tsunoda H, Teramoto R (2021) Antibody design using LSTM based deep generative model from phage display library for affinity maturation. Sci Rep 11 (1):5852. doi:10.1038/s41598-021-85274-7

Appendix

Summary

Over several decades, a variety of computational methods for drug discovery have been proposed and applied in practice. Traditionally, drug design is always taken as an effort of combinational optimization in discrete chemical space. Hence, optimization methods were exploited to search for new drug molecules to meet multiple objectives. With the accumulation of data and the development of machine learning methods, computational drug design methods have gradually shifted to a new paradigm. Especially, deep learning methods have attracted particular interest in drug design. In **chapter 2**, I give a brief description of two different *de novo* methods, compare their application scopes and discuss their possible development in the future.

In drug discovery, recurrent neural networks (RNNs) have been shown to be an effective method to generate novel chemical structures in the form of SMILES. However, ligands generated by current methods have so far provided relatively low diversity. In **Chapter 3**, a new method (***DrugEx***) was proposed to design *de novo* drug-like molecules. ***DrugEx*** is also an RNN model (generator) trained through a special exploration strategy integrated into reinforcement learning. As a case study we applied our method to design ligands against the adenosine A_{2A} receptor (A_{2A}AR). Through comparing the performance with other methods, it was proven that candidate molecules designed by ***DrugEx*** had a larger chemical diversity, and better covered the chemical space of known ligands compared to the state-of-the-art.

In order to address the issue of polypharmacology, the ***DrugEx*** algorithm was updated with multi-objective optimization to generate drug molecules towards more than one specific target. The concept of evolutionary algorithms was merged into ***DrugEx***. During the training loop, scores for all objectives provided by the *environment* are used to construct Pareto ranks of the generated molecules with GPU-accelerated non-dominated sorting and Tanimoto-based crowding distance algorithms. In **Chapter 4**, the results of its application demonstrated the generation of compounds with a diverse predicted selectivity profile

toward multiple targets, offering the potential of high efficacy and lower toxicity.

As the chemical space to search for feasible drug-like molecules is immense, rational drug design tends to start from known scaffolds as the pharmaceutical core to be optimized, e.g., add or modify substituents. In order to improve its generality, ***DrugEx*** was further updated to have the capability of designing molecules based on given scaffolds consisting of multiple fragments. In **Chapter 5**, we extended the architecture of Transformer to deal with each molecule as a graph. The encoder of the graph Transformer receives the input graph of scaffolds containing multiple fragments and its decoder outputs the graph-based molecule containing the given scaffolds. We trained this generator under the reinforcement learning framework to increase the number of active ligands. As a proof, our proposed methods compared with SMILES-based methods to design $A_{2A}AR$ ligands. The results demonstrated its effectiveness in that 100% valid molecules are generated and most of them had predicted high affinity towards $A_{2A}AR$ with given scaffolds.

Up to now, widespread adoption of new *de novo* drug design techniques in the field of drug discovery is still lagging behind the most recent developments. In order to establish a close collaboration between diverse groups of experimental and theoretical scientists, ***GenUI*** was developed as a visualizion software platform that makes it possible to integrate molecular generators within a feature-rich graphical user interface. ***GenUI*** is an open-source web service and ***DrugEx*** was integrated as a proof of principle. In **Chapter 6** the details of ***GenUI*** are presented to show how it facilitates collaboration in the disparate communities interested in computer-aided drug discovery.

In Chapter 7, I draw conclusions about my contributions to the development of computational drug design and provided my perspective on how AI can be further applied in drug discovery. With these studies I made to a comprehensive investigation of the application of cutting-edge AI methods to design *de novo* drug molecules for biological targets. These studies highlight the overwhelming power of AI methods in drug discovery.

Samenvatting

In de afgelopen decennia is een verscheidenheid aan computationele methoden voor het ontdekken van geneesmiddelen ontwikkeld en in de praktijk toegepast. Optimalisatiemethoden worden gebruikt om moleculen te zoeken die op meerdere biologische doelen effect hebben. Met de accumulatie van gegevens en de ontwikkeling van methoden voor machinaal leren, zijn computationele methoden voor het ontwerpen van geneesmiddelen geleidelijk verschoven naar een nieuw paradigma. Vooral deep learning methoden zijn interessant geworden voor het ontwerpen van geneesmiddelen. In hoofdstuk 2 geef ik een korte beschrijving van twee verschillende de novo methoden, vergelijk ik hun toepassingsgebieden en bespreek ik hun mogelijke ontwikkeling in de toekomst.

Bij het ontdekken van geneesmiddelen is aangetoond dat terugkerende neurale netwerken (RNNs) een effectieve methode zijn om nieuwe chemische structuren in de vorm van SMILES te genereren. Liganden die met de huidige methoden zijn gegenereerd, hebben tot nu toe echter een relatief lage diversiteit opgeleverd. In Hoofdstuk 3 wordt een nieuwe methode (***DrugEx***) voorgesteld om de novo moleculen te ontwerpen die lijken op medicijnen. ***DrugEx*** is ook een RNN-model (generator) dat is getraind via een speciale verkenningsstrategie die is geïntegreerd in ‘reinforcement learning’. Om te bewijzen dat onze methode werkt hebben we liganden ontworpen voor de A2A-receptor (A$_{2A}$AR). Door de prestaties te vergelijken met andere gelijkwaardige methoden, werd bewezen dat kandidaat-moleculen ontworpen door ***DrugEx*** een grotere chemische diversiteit hadden en beter de chemische ruimte van bekende liganden bedekten dan de andere methoden.

Om het probleem van polyfarmacologie aan te pakken, is het ***DrugEx***-algoritme bijgewerkt met multi-objectieve optimalisatie om moleculen te genereren voor meer dan één specifiek doelwit. Het concept van evolutionaire algoritmen werd samengevoegd met ***DrugEx***. Tijdens de training van het model worden scores voor alle doelstellingen van de omgeving gebruikt om Pareto-rangen van de gegenereerde moleculen te construeren. Dit wordt gedaan met GPU-versnelde niet-gedomineerde sortering en op Tanimoto gebaseerde algoritmen voor crowding-afstand. In Hoofdstuk 4 laten de resultaten moleculen met een divers voorspeld selectiviteitsprofiel naar meerdere doelwitten zien. Deze moleculen hebben een potentieel hoge werkzaamheid en lage toxiciteit.

Omdat het aantal potentieel geschikte moleculen met geneesmiddel-achtige kenmerken enorm is, start rationeel medicijn ontwerp vaak vanuit een al bekende farmaceutische structuur (‘scaffold’). Vervolgens worden dan bijvoorbeeld substituenten toegevoegd of vervangen. Om de algemeenheid te verbeteren, werd ***DrugEx*** verder geüpdatet om de optie te bieden moleculen te baseren op meerdere ingegeven scaffolds. In Hoofdstuk 5 breidden we de architectuur van Transformer uit om moleculen als een netwerk te representeren. De codeerder van de netwerk Transformer ontvangt het inputnetwerk van meerdere scaffolds en de decodeerder produceert het molecuul als netwerk dat de gegeven scaffolds bevat. We trainden de generator met het ‘reinforcement learning’ model om het aantal actieve liganden te vergrootten. Ter bewijs hebben wij onze voorgestelde methoden vergeleken met op SMILES gebaseerde methoden om $A_{2A}AR$ liganden te ontwerpen. De resultaten lieten een hoge effectiviteit zien. Er werden namelijk 100% valide moleculen gegenereerd en de meesten hiervan hadden hoge voorspelde affiniteit voor de $A_{2A}AR$ met de gegeven scaffolds.

Op dit moment is er nog geen sprake van een brede adoptie van nieuwe de novo drug design technieken voor de ontwikkeling van geneesmiddelen. Om een betere samenwerking tussen experimentele en theoretische wetenschappers op te zetten is ***GenUI*** ontwikkeld. ***GenUI*** is een open-source web service dat dient als visualisatieplatform voor moleculaire generatoren, met daarnaast veel verschillende functies in de grafische interface. Als bewijs van principe is DrugEx geïntegreerd in ***GenUI***. Hoofdstuk 6 laat de details zien omtrent ***GenUI*** en hoe deze software gebruikt kan worden om samenwerkingen te bevorderen tussen groepen die geïnteresseerd zijn in computer-geasissteerde geneesmiddelenontwikkeling.

In hoofdstuk 7 trek ik conclusies over mijn bijdragen aan de ontwikkeling van het veld van computationele geneesmiddelenontwikkeling en deel ik mijn kijk op toekomstige toepassingen van AI in geneesmiddelenonderzoek. Met deze studies heb ik bijgedragen aan een uitgebreid onderzoek naar de toepassing van de meest moderne AI methodes om de novo medicijnmoleculen te ontwerpen voor biologische doelen. Deze studies laten de enorme kracht zien van het gebruik van AI methodes in geneesmiddelenontwikkeling.

中文总结

近些年来，已经有多种多样的用于药物发现的计算方法被提出并在实践中应用。传统上，药物设计总是被视为离散的化学空间中的优化组合问题，即利用优化方法来寻找能满足多个目标新药分子。随着数据的积累和机器学习方法的发展，药物设计中的计算方法逐渐发展出了一种新的范式，即深度学习方法大量应用于药物设计之中。 在**第二章**中，我写了一篇综述来简要描述了两种不同种类方法，比较了它们的应用范围，并讨论了它们未来可能的发展。

在药物发现中，循环神经网络 (RNN) 已被证明是一种以 SMILES 形式生成新化学结构的有效方法。然而，迄今为止的方法产生的化合提供的多样性相对较低。在**第三章**中，我提出了一种设计新药物分子的新方法 (***DrugEx***)。***DrugEx*** 也是一个 RNN 模型（生成器），在强化学习中通过的特殊的探索策略进行训练。作为案例研究，我们应用该方法来设计针对腺苷 A_{2A} 受体 ($A_{2A}AR$) 的配体。通过与其他方法的性能比较证明， ***DrugEx*** 设计的候选分子具有更大的化学多样性，并且与现有的方法相比，能更好地覆盖了已知化合物的化学空间。

为了解决多靶点药理学问题，***DrugEx*** 算法更新为多目标优化，以生成针对多个特定目标的药物分子。在该方法中，进化算法的概念被整合到 ***DrugEx*** 中。在训练步骤中，预测器提供的所有目标的分数，并用于构建生成分子的 Pareto 排序。在这里，我们通过 GPU 加速来改进“非支配排序”（non-dominant sorting) 和基于 Tanimoto 的“拥挤距离” (cowding distance) 算法。在**第四章**中，其应用结果证明了生成的化合物对多个目标具有不同的预测选择性，具有高效和低毒性的潜力。

由于寻找可行的类药物分子的化学空间是巨大的，合理的药物设计往往从已知的骨架作为要优化的药物核心开始，例如添加或修改取侧链基团。为了提高其通用性，DrugEx 进一步更新，使其具有基于给定的包含多个分子片段的骨架来设计分子的能力。在**第五章**中，我们扩展了 Transformer 的架构，将每个分子作为图结构处理。在基于图的 Transformer 中，编码器接收包含多个片段的骨架作为输入，其解码器输出包含基于给定骨架的图结构的分子。另外，我们也在强化学习框架下训练这个

生成器，以增加活性配体的数量。我们将提出的方法与基于 SMILES 的方法通过设计 $A_{2A}AR$ 配体来比较其性能。结果证明，其生成的分子 100% 为效分子，并且其中绝大多数包含给定的骨架，并被预测为具有 $A_{2A}AR$ 高亲和力。

到目前为止，在药物发现领域广泛采用的药物设计技术仍然落后于最近的发展。为了在不同的实验和计算科学家群体之间建立密切合作，***GenUI*** 被开发为一个可视化软件平台，可以在功能丰富的图形用户界面中集成分子生成器。 ***GenUI*** 是一个开源网络服务，并且 ***DrugEx*** 被作为案例而整合在其中。在**第六章**中，我们详细地描述了 ***GenUI*** 各种细节，并用以展示如何促进对计算机辅助药物研发感兴趣的不同社区之间的协作。

在**第七章**中，我对关于我对计算药物设计发展的贡献进行总结，并提供对人工智能如何进一步应用于药物研发的看法。通过这些研究，我对如何应用最新的人工智能方法来为生物靶点设计新药分子做出了综合的研究。这些研究突出了人工智能方法在药物发现中的未来可能的决定性力量。

List of publications

Part of this thesis

Liu X, Ye K, van Vlijmen HWT, IJzerman AP, van Westen GJP. An exploration strategy improves the diversity of de novo ligands using deep reinforcement learning: a case for the adenosine A_{2A} receptor. Journal of cheminformatics. 2019;11 (1):35.

Liu X, IJzerman AP, van Westen GJP. Computational approaches for *de novo* drug design: past, present, and future. Methods Mol Biol. 2021;2190:139-65.

Liu X, Ye K, van Vlijmen HWT, Emmerich MTM, IJzerman AP, van Westen GJP. DrugEx v2: *de novo* design of drug molecules by Pareto-based multi-objective reinforcement learning in Polypharmacology. Journal of cheminformatics 2021:13(1):85.

Sicho M, **Liu X**, Svozil D, van Westen GJP. GenUI: interactive and extensible open source software platform for de novo molecular generation and cheminformatics. Journal of cheminformatics. 2021;13(1):73.

Liu X, Ye K, van Vlijmen HWT, IJzerman AP, van Westen GJP. DrugEx v3: Scaffold-Constrained Drug Design with Graph Transformer-based Reinforcement Learning. (*preprint*)

Not part of this thesis

Yang Z, Wang C, Wang T, Bai J, Zhao Y, **Liu X**, et al. Analysis of the reptile CD1 genes: evolutionary implications. Immunogenetics. 2015;67(5-6):337-46.

Liu X, Yang S, Li C, Zhang Z, Song J. SPAR: a random forest-based predictor for self-interacting proteins with fine-grained domain information. Amino Acids. 2016;48(7):1655-65.

Chen Z, **Liu X**, Li F, Li C, Marquez-Lago T, Leier A, et al. Large-scale comparative assessment of computational predictors for lysine post-translational modification sites. Brief Bioinform. 2019;20(6):2267-90.

Chen Z, He N, Huang Y, Qin WT, **Liu X*,** Li L. Integration of A Deep Learning Classifier with A Random Forest Approach for Predicting Malonylation Sites. Genomics Proteomics Bioinformatics. 2018;16(6):451-9..

Chen Z, **Liu X**, Zhao P and Song J. iFeature$^{\mathrm{Omega}}$: a comprehensive platform for generating, analyzing and visualizing various representations for biological sequences, structures and ligand. (*Submitted*)

Curriculum Vitae

Xuhan Liu was born on September 28th, 1989 in Jia County, Henan province, China. From 2007 to 2012, he studied in the College of Life Sciences in Henan Agricultural University. Besides following the majority of subjects in biological sciences, he also took a large amount of time in learning computer sciences and designed the first website for the biological lab where he took his internship. In the end, he obtained a Bachelor degree in biology with one year extension when he planned to further study the cross domain between computer sciences and biology.

In the following three years, he studied in the college of biology in China Agricultural University and was supervised by Prof. Dr. Ziding Zhang in the protein bioinformatics group. In this group, he obtained extensive knowledge of artificial intelligence. He also applied what he learned in the project he undertook about self-interacting protein prediction. In this project, he proposed a novel feature encoding scheme and successfully improved the performance of the predictive model. He also tried to apply deep learning with ambitious faith in spite of the failure in the end. In addition, he also designed a webserver to manage and release the information for this lab. In 2015, he obtained the Master degree in biology (bioinformatics).

From 2015 to 2017, he worked as a bioinformatician in Oriental YAMEI Gene Technology Research Institute Co., Ltd. His main work was the study of association between human genotype and phenotype under the lead of Prof. Dr. Li Shen. He developed the online office system for the company to generate and manage the gene detection reports of customers. Moreover, he also collaborated with Dr. Zhen Chen to study the prediction and functional analysis of lysine post-translational modification in proteins with deep learning.

After obtaining funding from China Scholarship Council in 2017, he began his study at the division of drug discovery and safety at LACDR/Leiden University under the supervision of Prof. Dr. A.P. IJzerman and Prof. Dr. G.J.P van Westen. In addition, he also spent time in the academic activities organized by Prof. Dr. Aske Plaat and Dr. Mike Preuss in the reinforcement learning group to learn cutting-edge technologies of AI in LIACS/Leiden University. During these four years study, he thoroughly investigated the application of AI in drug *de novo* design. Based on these studies, he proposed a method named ***DrugEx*** containing relevant Python-based algorithms and released it on GitHub to facilitate its accessibility in the scientific community. During these four years, he pitched his work during different public academic occasions. In the last three months, he also got an opportunity to take an internship at Janssen Pharmaceuticals, also as a possible preparation for his career in the future.

Acknowledgements

My four years of research at Leiden University were not a smooth journey but filled with failures and setbacks. Especially, the pandemic outbreak of COVID-19 led to being forced to work from home, which made me doubt myself whether I can complete my PhD studies. Thanks to my friends who made endless efforts to help me overcome the hardships, I persisted in completing these projects and finishing the writing of this thesis. I will therefore give my full gratitude to the following people.

First of all I would like to thank my promoter Prof. Ad IJzerman. I am indebted to your suggestions about scientific directions which guided me to move forward and I also learned from your academic integrity embodied in your attitude towards every details of scientific research. Besides supervision of my study you also gave me advice about how to adapt myself to the culture of the Netherlands. I am also grateful for the help of my daily supervisor Prof. Gerard van Westen, who organized the academic activities and managed the computational equipment to provide an excellent scientific environment in our group. You also listened to me and helped me in all aspects of the life as a PhD student. In addition, I also thank my co-promoter Prof. Kai Ye and Prof. Herman W. T. van Vlijmen, who gave me useful comments about my works regularly. In addition, Prof. van Vlijmen provided me with an opportunity to take an internship at Janssen Pharmaceuticals.

In the division of drug discovery and safety, there are many kind-hearted colleagues, who establish a joyful atmosphere. I thank Martin that we cooperated to integrate ***DrugEx*** into web-based ***GenUI*** and successfully guided bachelor students into computational drug design. I also thank my master student Milan who successfully proved the effectiveness of ***DrugEx*** for other GPCRs. For Olivier, Bart, Willem, Brandon, Lindsey, Marina and Martin, thank you for discussing with me and giving me help when I met problems. Interestingly, I still remember the discussion about the name of ***DrugEx*** with Marina as a nice officemate. Thank you, Roelof, to fix the bugs on the computational servers. Thanks to Lia and Rongfang for arranging the daily routine and inform me on time. For Huub, Bert and

Sebastian, it was an unforgettable moment to take part in the Leiden Science run with you at my thirtieth birthday.

I was lucky to have a chance to learn many new AI methodologies from the reinforcement learning group in LIACS. Thanks to Prof. Aske Plaat and Dr. Mike Preuss for organizing regular meetings to discuss the fast developments in AI technologies. Thanks also to Dr. Michael Emmerich who provided me with the solutions of multi-objective optimization. I also thank Hui Wang for discussing the detailed application of AI approaches. Moreover, I also thank Dr. Erik van Geest from LIC and Zhenzheng Zhang from LACDR for collaborating with each other to undertake the projects about graphene and metabolism, respectively. Specially, I thank Prof. Zhen Chen in Henan Agricultural University who offered the opportunity to apply deep learning methods in biomedicine for the first time.

In Leiden I have made many truly good friends. Special thanks go to Xue Yang with whom I discussed many psychological theories and applied them to solve the problems we met in real life. I thank Wei Chen, Wensen Li, Xiangyu Liu, Xiaoyue Song and Li Jia for jogging 5km almost every afternoon together with me, which was great to keep my health state. I also thank my friends Di Wang, Yixin Luo, Wei Wang, Zhao Yang, Jing Zhang, Zhuoyi Luo and Ruochen Oyang for preparing delicious food and organizing parties frequently. Without you, my life in this foreign country would not have been so wonderful.

Last but not least, I thank the selfless support from my parents and my wife Dr. Lily Wang despite the remote distance between us. Because of your unrequited love, I feel strong to make up my mind and to accomplish my scientific research alone but not lonely. No matter what kind of achievements are ahead of me, I am more than willing to share these with you forever.