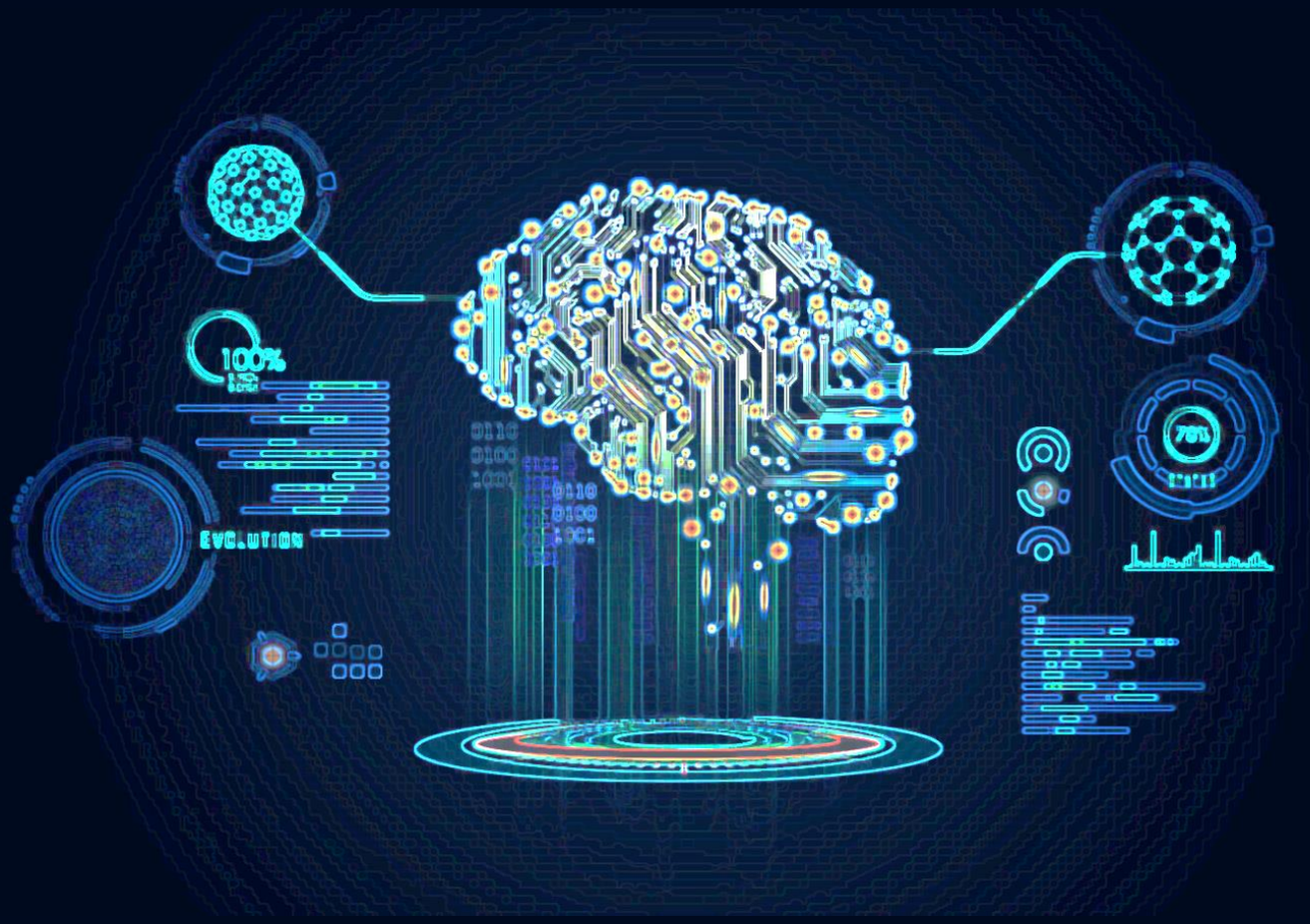
Chapter 2

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].

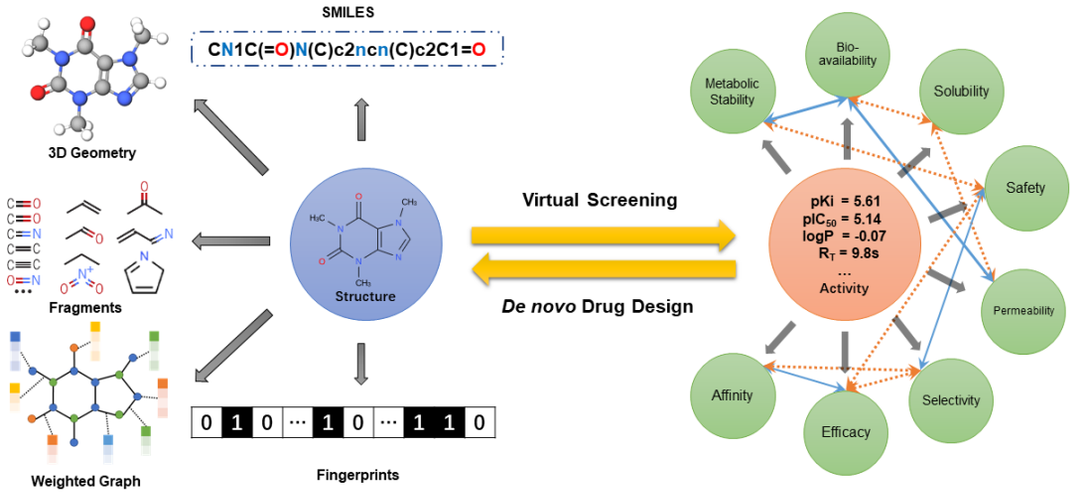


Fig. 2.1: Schematic overview of the interplay of two methods in computational drug discovery: virtual screening and *de novo* design. The left of the figure shows ways in which a molecule can be described for computational methods (see 'Molecular Representations'). On the right the multi-objective nature 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.

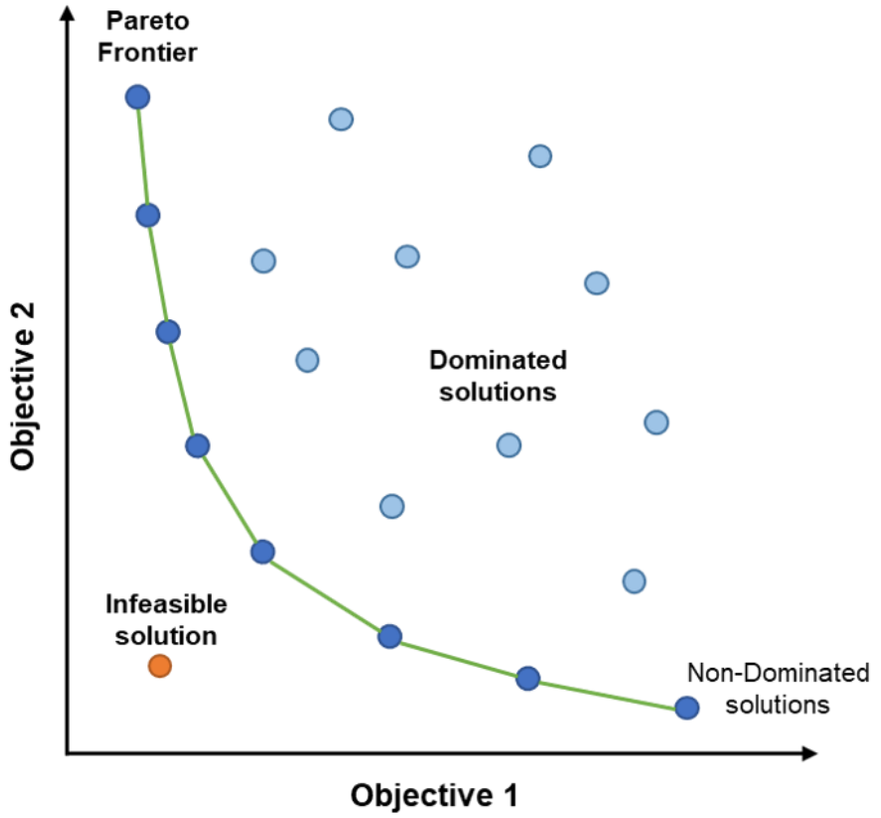


Fig. 2.2: Pareto frontier in multi-objective optimization. Take two objectives as an example, non-dominated solutions form a boundary called Pareto frontier which separates the infeasible solutions in the lower left region from dominated solutions in the upper right region.

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.

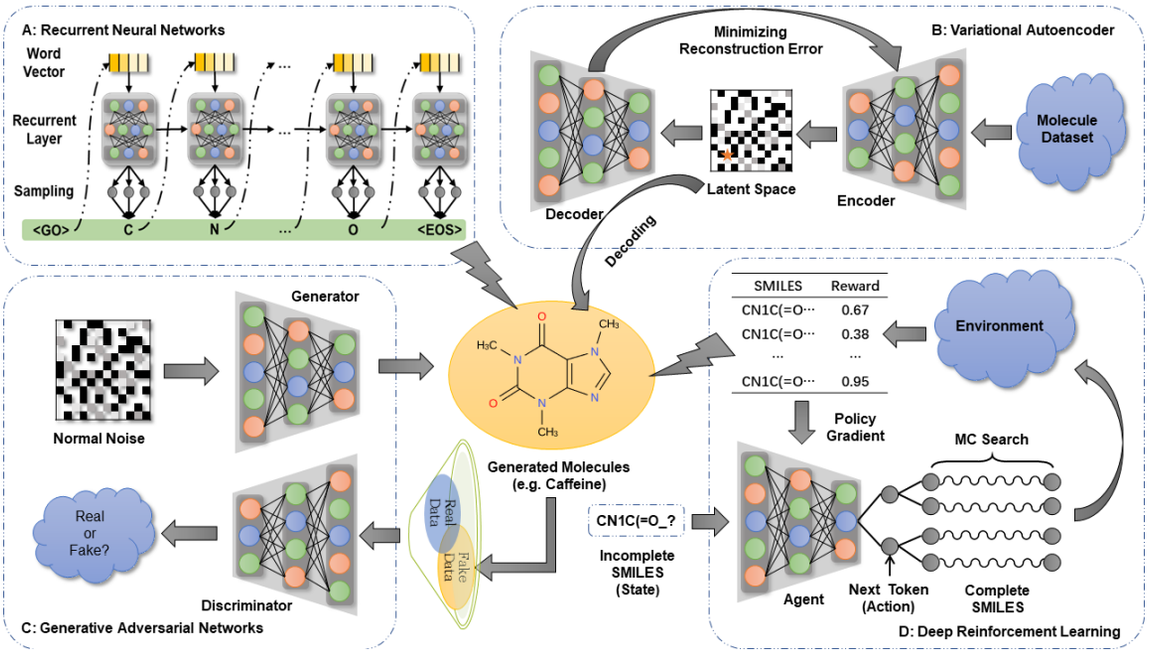


Fig. 2.3: Four basic deep learning architectures commonly used in de novo drug design, including recurrent neural networks (A), variational autoencoder (B), generative adversarial networks (C) and deep reinforcement learning (D).

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:

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

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

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

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 $\boldsymbol{p(x)}$ and is also rewritten as:

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

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

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

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

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 $\mathbf{s_i} = [\boldsymbol{a_1}, ..., \boldsymbol{a_{i-1}}]$. 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:

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

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,

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

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:

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

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:

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

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

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:

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

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

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

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].

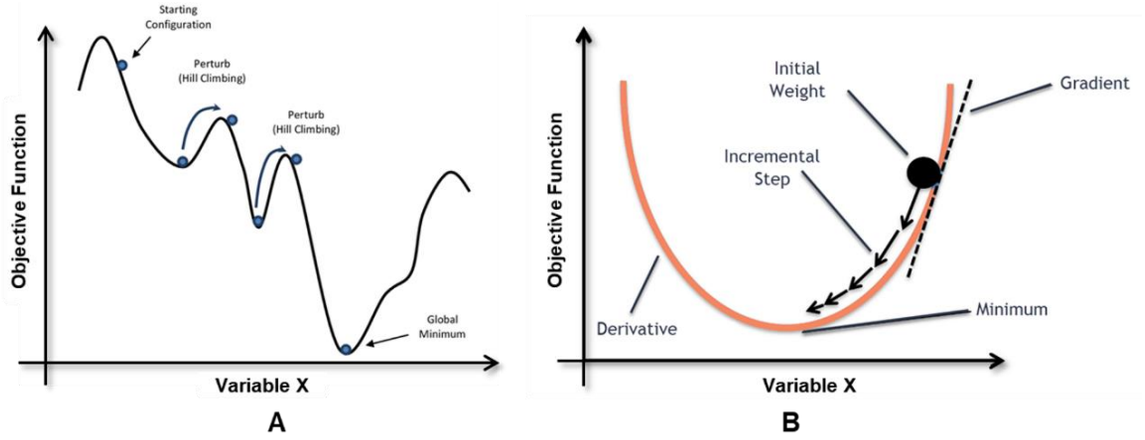


Fig. 2.4: Objective functions for optimization methods (A) and deep learning methods (B). Usually, objective functions in optimization methods contain many local minima/maxima, while non-convex objective functions (also called loss functions) are deliberately constructed in deep learning methods to make 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