Deep reinforcement learning for multiparameter optimization in *de novo* drug design

Niclas Ståhl,[†*] Göran Falkman,[†] Alexander Karlsson,[†] Gunnar Mathiason[†] and Jonas Boström[‡]

†School of Informatics, University of Skövde, Skövde, Sweden. ‡Medicinal Chemistry, Early Cardiovascular, Renal and Metabolism, R&D BioPharmaceuticals, AstraZeneca, Gothenburg, Sweden

Corresponding author e-mail: niclas.stahl@his.se

Abstract

In medicinal chemistry programs it is key to design and make compounds that are efficacious and safe. This is a long, complex and difficult multi-parameter optimization process, often including several properties with orthogonal trends. New methods for the automated design of compounds against profiles of multiple properties are thus of great value. Here we present a fragment-based reinforcement learning approach based on an actor-critic model, for the generation of novel molecules with optimal properties. The actor and the critic are both modelled with bidirectional long short-term memory (LSTM) networks. The AI method learns how to generate new compounds with desired properties by starting from an initial set of lead molecules and then improve these by replacing some of their fragments. A balanced binary tree based on the similarity of fragments is used in the generative process to bias the output towards structurally similar molecules. The method is demonstrated by a case study showing that 93% of the generated molecules are chemically valid, and a third satisfy the targeted objectives, while there were none in the initial set.

Introduction

The goal in drug discovery is to identify new compounds with desirable properties suitable to become an approved drug. The process is long and demanding, both in terms of money and time. There is a large risk (>90%) that a promising compound fails in clinical trials, resulting in needlessly spent resources. The quest for discovering a small-molecule drug is non-trivial and the chemical space to search is mind-blowingly vast.[1] Hence, it is an infeasible task to screen all these molecules and subsequently select the molecules with the best properties. The search must therefore be concentrated to smaller subsets of compounds that are promising according to some heuristics. One way to approach that problem is to perform a directed search to identify novel compounds which fulfil a set of relevant criteria and to, for example, focus on the chemical space close to known actives. This well-understood approach is based on the premise that similar molecules exhibit similar behavior.[2] An example of this is the opioids morphine, codeine and heroin. These three drugs have very similar structure and are all, to some extent, used to treat pain. There are plenty of cases where natural product scaffolds have served as leads to drugs. Indeed, designing and making structurally similar compounds can be a very successful strategy. For example, there are numerous so-called "me-too" drugs, where as many as 70% of the pairs of me-too drugs are structurally very similar.[3] Moreover, it was recently shown that a significant fraction of candidate drugs is structurally close to their corresponding lead molecule.[4] Despite past successes, there seems to be a limited interest on how to confine the search space of *in silico* design methods. Instead, many recent approaches focus on "novelty", and how to design structurally "diverse" compounds. These terms are non-trivial to define, and the rationale for this behavior is probably driven by assumed advantages with respect to intellectual property (IP). However, following the advice from Murcko,[5] great medicinal chemists "*do not worry over IP*" in the discovery phase, but stay on top of the competitive landscape. In addition, given the current state of computer-aided molecular design (*i.e.* we cannot make perfect

prospective predictions), it is often an advantage to start projects using known active compounds, if there are any.[4]

Current strategies to constrain the search space predominantly lie in virtual screening of chemical libraries.[6] These may be very large virtual libraries, generated from available chemical reagents.[7,8] An issue with this approach is that the search is then limited to molecules that can be constructed from the initial set of reagents. And, obviously, it is not possible to find something that is not there. Hence, the outcome of a virtual screening is very dependent on the library size and how it was constructed. In a virtual screen, molecules are filtered successively at an increasing computational cost, using for example, ultrafast shape-based similarity tools[9] or machine learning methods than can estimate properties of the targeted molecules. There is a large variety of machine learning methods being used for this purpose, examples include k-nearest neighbors, random forest and artificial neural networks.[10–12]

The term *de novo* design describes the application of computational methods to generate new compounds in an automated fashion.[13] One such approach is to identify promising compounds through inverse design.[14] Here the desired molecular properties are specified *a priori,* and the aim is to generate compounds that fit the description. This has been considered a very difficult problem in the past. But, thanks to advances within the field of machine learning and artificial intelligence the approach is becoming popular. This type of approach is typically either based on evolutionary algorithms[15] or reinforcement learning algorithms.[16] Olivecrona *et al.*[17] and Popova *et al.*[18], for example, use the REINFORCE algorithm[19] to generate new molecules by the generation of SMILES strings. The SMILES concept of representing molecules as a line notation, similar to natural language, is as simple as elegant.[20] The power of using fragmented SMILES strings for similarity searching has been demonstrated with the text-based molecular LINGO approach,[20] just to name one example. In inverse design,[16-18] new SMILES strings are generated from an underlying distribution of

characters, where the probability distribution of the next character in the string is given by the previous characters. This roughly corresponds to generating molecules atom by atom. With this approach, it is generally claimed that models can generate >90% valid molecules and bias the generation towards a given property. However, this type of generative methods requires that the model is trained in advance against very large databases. Besides taking heaps of compute resources (*i.e.* a long time) to train such a model, it assumes that molecules with similar SMILES strings have similar properties. Thus, it can introduce a bias towards specific ways of constructing SMILES strings (e.g. canonical SMILES). This can be unfortunate since two molecules that have similar structure can have multiple completely different SMILES strings. Also, since new molecules are generated through a sampling process there is a risk for an unlucky event (brackets or numbers placed in an erroneous way) that may make the generated molecule not valid. Generative models therefore favor safer sampling steps, often causing long carbon chains and other easy-to-generate structures to be part of the final molecules. A few issues with using SMILES string in generative models have been revealed. Arús-Pous *et al.* recently demonstrated issues due to the SMILES syntax with respect to the ability to generate molecules with many rings and heteroatoms.[22] Slightly more serious is that such contemporary methods generate SMILES strings lacking chiral information,[17] and thus neglect the significance of stereoisomers in drug design.[23] To alleviate these problems, promising work is on-going to develop improved string representations of molecules. One such attempt is due to O'Boyle and Dalke who created the DeepSMILES representation,[24] which does not produce the same problem with brackets and rings as the regular SMILES when generating new molecules character by character.

Inverse design is not limited to methods working on SMILES strings. There are several approaches that work directly on the underlying graph structure of the molecule. This is well-explained in the review by Elton *et al.*,[25] who describe the current state of the art and note that there currently is a shift away from SMILES strings towards more sophisticated

representations. Gómez-Bombarelli *et al.*[26] and Kang and Cho,[27] for example, apply variational auto-encoders (VAE)[28] on graphs representing the structure of molecules to generate drug-like molecules. These works aim to find a bi-directional mapping between molecules and a latent space, representing some abstract features of the molecule. Small changes can then be applied to the latent representation of the molecule, which then would map to another molecule with similar latent features. Another deep learning approach is by Li *et al.*,[29] which uses a generative adversarial network (GAN).[30] In this approach, one model is used to generate molecules while the other is used to discriminate between generated molecules and molecules from a given dataset. The two models compete against each other, and both must improve to win over the other. The assumption is that the generative model will win and be able to generate novel molecules that are largely indistinguishable from the ones in the original dataset. However, it has been shown that GANs are difficult to train and that the distribution of samples from the generative model is unstable and easily collapse.[31,32] A third approach that also works directly on graph structures and searches for novel compounds close to existing ones has recently been presented by Li *et al*.[29] They use several different deep learning models to predict beneficial modifications to a given model. Methods exploiting molecular conformations and 3D shapes are also underway.[33] However, it has proven problematic to fully realize advantages of 3D based techniques, mainly due to the difficulty of obtaining accurate descriptions of molecules bioactive conformations.

In this paper, we present a reinforcement learning approach that is based on an actor-critic model for the generation of novel molecules. The model learns how to modify and improve molecules for them to have desired properties. The approach therefore distinguishes itself from previous reinforcement learning approaches in that it focuses on how to generate novel compounds structurally close to existing ones by transforming fragments in lead molecules. Thus, the presented approach does not attempt to search the entire chemical space to find optimal drug candidates. Instead, it focuses the search on

molecules structurally similar to lead molecules, with desirable "sweet spot" properties. Even though it may look like such an approach would be very limited, and only a few unique molecules could be created, this is not necessarily the case. Since every fragment can be replaced with several similar fragments and each molecule consists of multiple fragments there is an abundance of combinations. Say, if a molecule is split up into six fragments and each fragment is replaced with ten similar fragments, it is then possible to generate almost two million unique molecules with this approach, which is comparable to the size of Big Pharma high-throughput screening (HTS) collections.[34] Because the search starts from an initial set of molecules and is expanded around these, it means that the number of searched molecules will be smaller than in a modern virtual screening approach,[7,8] but the library of fragments can be much larger since the generation of new potential candidates is conducted in a smart way (e.g. it remembers which bonds were broken, and uses that information when piecing together new molecules), and far from all candidates are investigated. Hence, our approach uses inverse design, but uses a fragment library when generating molecules and only focus on molecules that are structurally close to known lead molecules.

Deep reinforcement learning

The methods presented in this paper are based on deep reinforcement learning (RL), which is further described in this section. Within the reinforcement learning framework is an agent that interacts with molecules and replaces their fragments with other similar fragments. The actions of this agent are controlled by a recurrent neural network, and this kind of network is described in the next section. Our work distinguishes itself from previous works by using an actor-critic approach. This is high-lighted in the end of this section, where actor-critic models are described.

Recurrent neural networks

A recurrent neural network (RNN)[35] is a special type of artificial neural networks where there are cyclic connections between neurons, unlike basic feed forward networks that are acyclic. Cyclic connections enable the network to have an inner representation of the current state. This gives the network the ability to remember information from previous steps in a sequence, which makes it advantageous to use RNNs for the analysis of sequential data, such as text or a molecule represented as a sequence of fragments.

RNNs work in a sequential fashion and process one step in a series at a time. Standard RNN networks can therefore only take decisions based on previous steps in the sequence that is analyzed. This shortcoming can, however, be resolved by using bidirectional networks.[36] A bidirectional network consists of two separate RNNs, in which one network analyses a sequence from the start to the end, while another network analyses the sequence in the opposite direction. This approach makes it possible for an RNN to take decisions based on information of previous and subsequent steps in a sequence.

One drawback with standard RNNs is that they have difficulties in capturing long-range dependencies, due to vanishing gradients.[37] However, this problem has partially been solved by the introduction of long short-term memory (LSTM) cells.[38] These cells allow the network to keep an inner state that the network can access, reset and update. Such networks are therefore more stable and can capture dependencies that are separated by a greater number of steps in the analyzed sequence.

Reinforcement learning

Reinforcement learning (RL) is a sub-field of machine learning concerned with how agents take actions in an environment to maximize some notion of reward. We consider the standard reinforcement learning framework,[39] in which an agent, with the capability to learn, interacts with an environment. This process is described by a discrete-time Markov decision process (MDP) with finite numbers of actions and states, and finite rewards. The set of

possible states in the MDP is denoted by S and the set of possible actions is denoted A. The state, action and reward at each time step t in the MDP are denoted s_t, a_t and r_t, respectively, where $s_t \in S$ and $a_t \in A$. The action a_t that the agent selects at time t is decided using the policy π, where $\pi(a, s) = P(a_t = a \mid s_t = s)$. The aim of the agent is to learn an optimal policy π^*, which accumulates the maximum reward over time. To find such a policy, we consider the expected value for an agent, following policy π, to be in the state s. This is given by the value function:

$$V^{\pi}(s) = \mathbb{E}^{\pi}[r_{t+1} + \gamma r_{t+2} + \gamma_2 r_{t+3} + \ldots \mid s_t = s], \quad \text{Eq. 1}$$

$$= \mathbb{E}^{\pi}[r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s], \quad \text{Eq. 2}$$

where γ is a discount factor making rewards in near time more desirable than those in the distant future. The value of taking a given action a in state s is defined in a similar way:

$$Q^{\pi}(s,a) = \mathbb{E}^{\pi}[r_{t+1} + \gamma r_{t+2} + \gamma_2 r_{t+3} + \ldots | s_t = s, a_t = a], \quad \text{Eq. 3}$$

$$= \mathbb{E}^{\pi}[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) | s_t = s, a_t = a] \quad \text{Eq. 4}$$

Most previous work using RL for the inverse design of molecules,[17-18] uses the REINFORCE algorithm[19] to find an optimal strategy on how to generate new molecules. This algorithm uses a differentiable policy, parameterized by θ. To measure how good the selected policy is, a policy score function that calculates the expected reward of a policy with parameter θ is needed. Using the value function, defined in Equation 1, such a function can be defined as:

$$J(\theta) = \sum_{\{s \in S\}} d^{\pi}(s) V^{\pi}(s) \quad \text{Eq. 5}$$

where d^{π} is the stationary distribution of states under the policy π. The target of finding the optimal policy π^* can now be solved through maximizing equation (Eq. 5) through gradient ascent where:

$$\nabla_\theta J(\theta) = \mathbb{E}^\pi [\nabla_\theta \pi(s,a)\, Q^\pi(s,a)]. \quad \text{Eq. 6}$$

One problem with this approach is that the function $Q^\pi(s,a)$ must be found, and then re-evaluated for each improved policy. While it is infeasible to find the real value of $Q^\pi(s,a)$, an unbiased sample can be collected by letting the agent start from state s_1 and then taking actions according to the policy. The total reward achieved by the agent can then be used as an unbiased estimate of $Q^\pi(s,a)$. However, even though the policy eventually will converge towards the optimum with this strategy, it may require a large computational effort due to the large variance in the samples of $Q^\pi(s,a)$.

Actor-critic models

Actor-critic models make use of temporal difference (TD) learning,[40] by having a critic model that evaluates the behavior of the agent and suggests whether the agent performed well or not. To achieve this, the critic uses bootstrapping to approximate the value function $V(s)$, introduced in Equation 1. Hence, the agent interacts with the environment and learns in the same way as in the REINFORCE algorithm, but the critic's approximation of the value function is used instead of a sampled trace to estimate $Q^\pi(s,a)$, reducing the variance in the training process and accelerating the learning of the agent. The critic's approximation of $V(s)$ is parameterized with w and, thus, a good approximation can be achieved by finding an optimal configuration of w so that the TD error (Eq. 7), is minimized:

$$(V(s_t) - (r_t + \gamma V(s_{t+1})))^2. \quad \text{Eq. 7}$$

Note that this makes actor-critic models fully online learners that learn instantly from each action taken and state visited. Hence, there is no need to keep track of traces from the MDP, and actions and rewards are only processed as they occur and then never revisited.

Method

The method presented here, which we name DeepFMPO, is based on an actor-critic model for reinforcement learning. This AI model learns how to modify compounds and improve them to have desired properties. To modify molecules, they must be encoded into a format that can be easily handled by the agent. The first step of the encoding is to split molecules into fragments. These fragments are then encoded into a sequence of ones and zeros (binary strings), so that similar fragments get similar encodings. The actor-critic model is then applied to these fragments to learn which of them that should be replaced to obtain more desirable properties. These steps are described in detail below.

Fragmentation

A library of fragments must first be generated. This library is built up by fragmenting a set of molecules. This set would typically consist of molecules that have showed activity against the biological target(s) of interest in the drug hunting project. But, it can be any data set, such as FDA approved drugs, commercially available chemical reagents, corporate screening collections, or various combinations thereof. A common way to fragment molecules is to split them up into classes such as R-groups, scaffolds and linkers.[41] We essentially follow the same scheme when splitting the molecules, but we do not sort the fragments into classes. Hence, all fragments are therefore treated the same. To fragment a molecule, all single bonds extending from a ring atom are broken. The attachment points are recorded and stored, for later use in the assembly step. The methods allows fragments with differing number of attachment points to be exchanged, provided that the total number of attachment points remains unchanged. The open source cheminformatics python software RDKit[42] was used throughout in this process. Although the fragmentation step does not allow ring-bonds to be broken, the subsequent assembly step allows for rings to be replaced by open-chains, and vice versa. Fragments with more than 12 heavy atoms are discarded in the process, as well

as fragments with four or more attachment points. These constraints are enforced to reduce the complexity while still being able to generate a multitude of interesting candidates. The fragmentation process is illustrated in Figure 1.

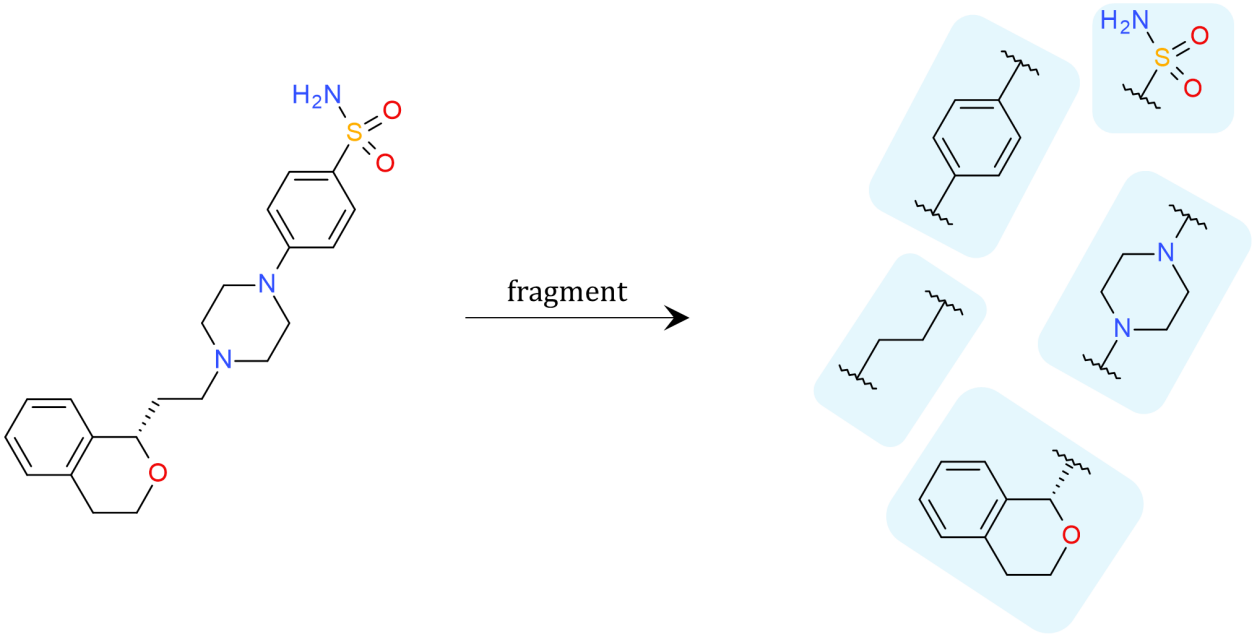


Figure 1. An illustration of the fragmentation process using the selective D4 receptor antagonist Sonepiprazole. Fragments are generated by breaking all single bonds between a ring atom and its connected atom.

Fragment encoding and similarity calculations

In our method all fragments are encoded as binary strings, and the aim of the encoding is that similar fragments should get similar encodings. The similarity between fragments must therefore be measured. There are many approaches to calculate chemical similarities.[43] A molecular fingerprint is an immediate binary encoding where similar molecules should, in principle, give similar encodings. However, when comparing fragments of molecules, with their inherent sparse representations, we found them to be less useful for this purpose (see Table 1). A chemically intuitive method to measure the similarity between molecules is to use the maximum common substructure *Tanimoto*$_{MCS}$ *(*T_S*)* similarity:[44]

$$T_s(M_1, M_2) = \frac{mcs(M_1, M_2)}{atoms(M_1) + atoms(M_2) - mcs(M_1, M_2)}. \quad \text{Eq. 8}$$

Here, $mcs(M_1,M_2)$ is the number of atoms in the maximum common substructure of molecules M_1 and M_2 and $atoms(M_1)$ and $atoms(M_2)$ is the number of atoms in molecule M_1 and M_2, respectively.

One advantage of the $Tanimoto_{MCS}$ similarity (Eq. 8), is that it directly compares the structures of the fragments and is therefore not dependent on other specific representations. This method generally works well when comparing "drug-like" molecules. There are, however, drawbacks of using the $Tanimoto_{MCS}$ similarity for smaller fragments. With this measure, fragments like methoxyethane "COCC" and butane "CCCC" will have a low similarity even if they only differ with one heavy atom (Table 1). That is, there is a general problem with this measure when comparing similar molecules that differ in the central part.

An alternative, which overcomes these shortcomings, is to calculate the similarity between SMILES strings. A common way to measure similarity between two text strings is to use the Levenshtein distance.[45] The Levenshtein distance is defined as the minimal number of insertions, deletions and replacements that are needed to make two strings identical. The Levenshtein distance between two strings, $L_s(s_1,s_2)$ is defined as $lev_{s_1,s_2}(|s_1|,|s_2|)$, where $|s|$ is the length of s and the function lev_{s_1,s_2} is defined as:

$$lev_{s_1,s_2}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} lev_{s_1,s_2}(i-1,j) + 1 \\ lev_{s_1,s_2}(i,j-1) + 1 \\ lev_{s_1,s_2}(i+1,j+1) + f_{s_1 \neq s_2}(i.j) \end{cases} & \text{otherwise} \end{cases} \quad \text{Eq. 9}$$

where $f_{s_1 \neq s_2}(i,j)$ is an indicator function, which is equal to 1 if character i of s_1 is not equal to character j of s_2 and 0 otherwise. The similarity of two molecules, which are represented by the SMILES strings s_1 and s_2, can then be measured as:

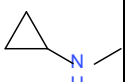
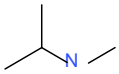
$$1 - \eta * L_s(s_1,s_2), \qquad \text{Eq. 10}$$

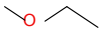
where η is a free parameter. In all experiments described in this paper, η is set to the value 0.1. This value is selected to make the $Tanimoto_{MCS}$ similarity (Eq. 8), roughly the same as the Levenshtein distance (Eq. 10), for medium sized fragments. This way of measuring similarities performs well for fragments of molecules. But, there are cases where it works less well. For example, when comparing the two structurally similar compounds *N*-methylpropan-2-amine "CC(C)NC" and its corresponding ring-closed analog *N*-methylcyclopropanamine "CNC1CC1" a low Levenshtein distance is obtained (Table 1). As a compromise, we choose to measure the similarity between two molecules M_1 and M_2, with the corresponding SMILES representations S_1 and S_2, as:

$$\max(T_s(M_1,M_2), L_s(S_1,S_2)). \qquad \text{Eq. 11}$$

Hence, the metric that presents the highest similarity between the pairs of molecules compared is used. In this fashion our approach favors similar fragments. Hence, we would like to couple fragments that show high similarity, independent of the outcome of the two similarity metrics used.

Table 1. $Tanimoto_{MCS}$ and Levenshtein distances for two pairs of small molecules illustrating strengths and weaknesses for both. Fingerprint similarities[a] ($Tanimoto_{FP}$) are shown for completeness.

No	*Structure*	*Structure*	$Tanimoto_{MCS}$ (T_s)	*Levenshtein*(L_s)	$Tanimoto_{FP}$[a]	*max*(T_s, L_s)
1			1.00	0.50	0.14	1.00

2			0.33	0.90	0.29	0.90

[a]RDKit's Morgan fingerprint with radius 2 and 2048 bits.[42]

Encoding fragments

All fragments are encoded into binary strings. These strings are created by constructing a balanced binary tree that is based on the similarity of the fragments. Subsequently, this tree is used to generate binary strings for each fragment and thus, in extension a binary string representing the molecule. The order of attachment points is considered as an identifier for each fragment. Hence, there may be several configurations of the same fragment in the tree. However, the combination of fragment structure and the order of attachment points are always unique and thus, can only exist in a single leaf of the tree.

When assembling the tree, similarities between all fragments are calculated according to Equation 11. Pairs of fragments are then formed in a greedy bottom-up manner, where the two most similar fragments are paired first. The process is then repeated and the two pairs with the most similar fragments are joined into a new tree with four leaves. The calculated similarity (Eq. 11) between two sub-trees is measured as the maximum similarity between any two fragments of these trees. The joining process is repeated until all fragments are joined together in a single tree.

When every fragment has been stored in the binary tree it can be used to generate encodings for all fragments. The paths from the root to the leaves, where the fragments are stored, determine the encoding of each fragment. For every branch in the tree a one (“1”) is appended to the encoding if going to the left and a zero (“0”) is added if going to the right, see Figure 2. Hence, the rightmost character in the encoding corresponds to the branching closest to the fragment.

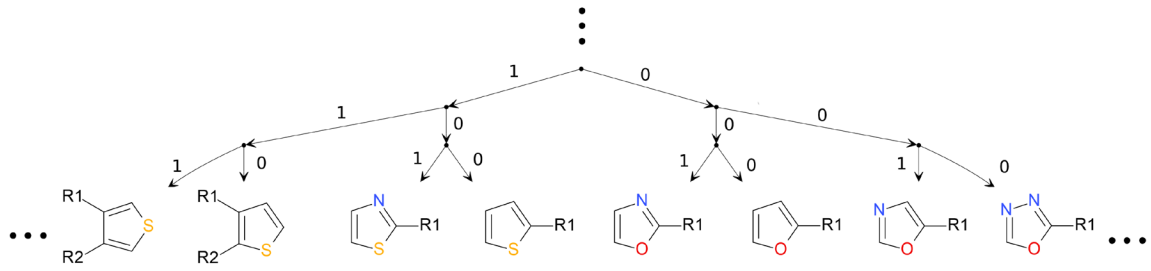


Figure 2. A sub-part of the binary tree holding all fragments. It can be seen that fragments that are similar to each other are placed close to each other in the tree. The encoding of a fragment is determined by the path from the root to the leaf where the fragment is stored. Every branching to the left will add a 1 to the end of the encoding and every branching to the right will add a 0. The attachment points of the fragments are marked with an R and the number of the order in which they will bind.

Generating molecules

As previously described, an actor-critic model is used to modify the fragmented molecules. The modifications are conducted by selecting a single fragment of the molecule and one bit in the representation of that fragment. The value in this bit is then swapped. That is, if it was 0 it becomes 1 and vice versa. This allows keeping track of the degree of change applied to the molecule, since modifying a bit at the end of the encoding would represent a change to a very similar fragment while a change in the beginning would represent a change to a very different type of fragment. The leading bits of the encoding will be kept fixed, and thus the model is only allowed to change bits at the end, to force the model to search only for molecules in the vicinity of already known compounds. In this context it should be noted that we define a valid molecule as a molecule that is not rejected by RDKit's molecule sanitizer.[42] Thus, the atoms in the generated molecule are all in common valences and charge states.

Experiments

To highlight the usability of the presented method we conducted an experiment aiming to identify novel candidate compounds with properties in a pre-defined "sweet spot". This

setup was designed to mimic a real-world drug design problem where a set of lead compounds are optimized towards a sweet spot through a multi-objective optimization process.[46] In these experiments we optimized towards three calculated properties, namely clogP,[47] polar surface area (PSA)[42] and molecular weight (MW). Lipophilicity (clogP) is a cardinal property in drug discovery and is often described as of utmost importance. For example, lipophilic compounds are likely to be rapidly metabolized, to show low solubility and to display poor oral absorption, which often impacts bioavailability. PSA have been used as a medicinal chemistry metric for the optimization of compounds ability to permeate cells, and molecular weight (MW) is also often considered important for various reasons. These three properties were also selected for a practical reason, as the RDKit framework[42] has methods to calculate them, thus making it easier for others to reproduce our experiment. However, the presented method is modular, and any model or criteria can be used in the optimization of properties. For example, predictive DMPK models for properties like solubility, permeability and clearance, as well as methods for synthetic feasibility,[48] can readily be plugged in to facilitate speedy progression towards project defined candidate drug target profiles. In this context it should be mentioned that C-lab[49] – the AstraZeneca's in-house property prediction service has been integrated in the DeepFMPO framework.

In all our experiments, we let both the actor and the critic in the actor-critic model be represented by two bidirectional LSTM-networks. The agent is trained to replace fragments of molecules, with the aim of achieving optimal properties, with the final goal being the generation of compounds in the defined sweet spot.

Data Set

In this proof of concept study, a data set of 15 836 molecules, reported to be potent at the dopamine D2 and D4 receptors, were extracted from the ChEMBL database (version 24).[50] These molecules were fragmented and stored as a balanced binary tree library for later use

in the building process. A subset of 387 potent dopamine D4 compounds which do not fulfil the sweet spot criteria were extracted to serve as the initial set of lead molecules. The data set contains a few different structural series, all including a tertiary nitrogen ionized at physiological pH essential for dopamine D4 binding.[51] The number of compounds (387) is close to what a typical drug hunting program would have access to. It should be noted that there is nothing in our experimental setup which would prevent the use of any other dataset, either for the generation of the library or for the set of lead molecules.

Optimization of targets and rewards

One major challenge in drug discovery is to design molecules optimized towards multiple properties, which may not correlate well. To show that the presented approach can handle such cases, three different properties, which can characterize the feasibility for a molecule to be suitable as a drug were chosen. The aim of the model is to be biased towards molecules with all three properties in its generation. That is, to produce molecules in the targeted "sweet spot". As mentioned above, the selected properties were MW, clogP and PSA. For these properties, the arbitrarily target ranges used are shown in Table 2.

Table 2. Targeted molecular properties and their desired value ranges.

Property	Minimal	Maximal
MW	320	420
clogP	2.0	3.0
PSA	40	60

The agent in the reinforcement learning framework was rewarded for every valid molecule it produced and got a higher reward if it managed to generate molecules with properties in the targeted ranges. The reward for fulfilling a given property is negatively correlated to how difficult it is for the agent to find such compounds and the reward at epoch *e* is given by:

$$r_i = \sum_p check(m_i, p) * \frac{1}{1 + d_{e,p}}, \quad \text{Eq. 12}$$

where *check*(m_i,*p*) is an indicator function that has the value of 1 when molecule m_i is in the targeted range for property *p* and 0 otherwise. A dynamically updated approximation ($d_{e,p}$) of the distribution of generated molecules that will fulfil the target for property *p* at epoch *e* is given by:

$$d_{e+1,p} \leftarrow (1 - \beta) * d_{e,p} + \beta \frac{n}{\sum_{i=0}^{n} check(m_i, p)}. \quad \text{Eq. 13}$$

The initial value of $d_{e,p}$ is set to the distribution of molecules that fulfil the targeted property ranges in the initial set and β defines how quickly the expected distribution of molecules changes. Having such a dynamically updated reward allows the agent to adapt, and not only target properties that are easily satisfied, while β prevents the reward to fluctuating too much. The value of β is set to 0.5 in this experiment.

Model architectures

Both the actor and the critic are modeled with a bidirectional LSTM network. The state at each time step consists of the structure of the current molecule and the number of steps that are left until the terminating state. These networks read the encoded molecules, fragment by fragment, in both directions. Hence, the input size to these networks is the same size as the number of bits required to represent the different fragments. As previously mentioned, the task of the agent network is to predict which fragment and bit should be selected. The size of the output of the agent network is therefore the same as the number of fragments multiplied by the number of bits that can be modified. Since this study mainly aims at showing the benefits of a fragment based RL approach we decided to use the commonly used and simple architecture of three hidden layers between the input and the output in the agent network:

the first of the hidden layers has 128 neurons, the second layer consists of 64 bidirectional LSTM cells and the final hidden layer has 32 neurons. The critic network is built up in the same way with the single difference that the output layer only has one output, since the critic only predicts the value of a given state. The ADAM algorithm,[52] with a learning rate of 0.0001, was used to find an optimal configuration of the parameters of the properties in both networks. The presented models are implemented in Keras 2.2.4 and Theano 1.0.3 is used as backend.[53,54]

Results

The presented model generated 33% "sweet spot" molecules which thus had all properties within the desired ranges. Beside this achievement, the model also managed to generate 93% unique molecules with a valid structure. The evolution of the percentage of generated molecules that demonstrate properties within the target ranges during the training process is shown in Figure 3. The distribution of the targeted properties among the molecules that are generated in the last 10 epochs as compared to the distribution of the properties in the initial set is shown in Figure 4. The percentage of molecules that have these properties in the "sweet spot" ranges are summarized in Table 3. Examples of initial lead molecules and their corresponding optimized AI-generated partners are shown in Table 4.

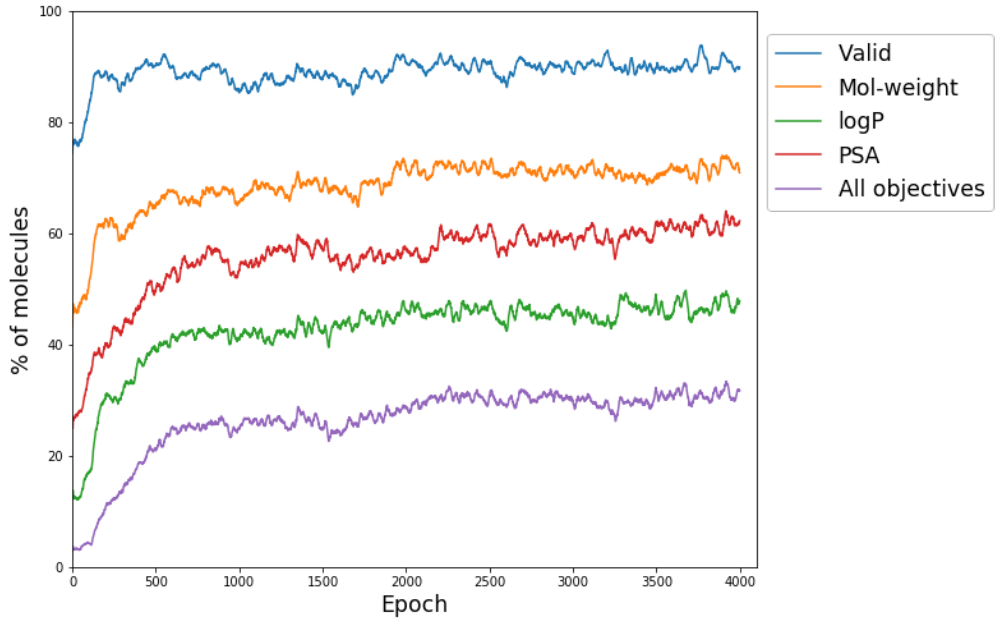
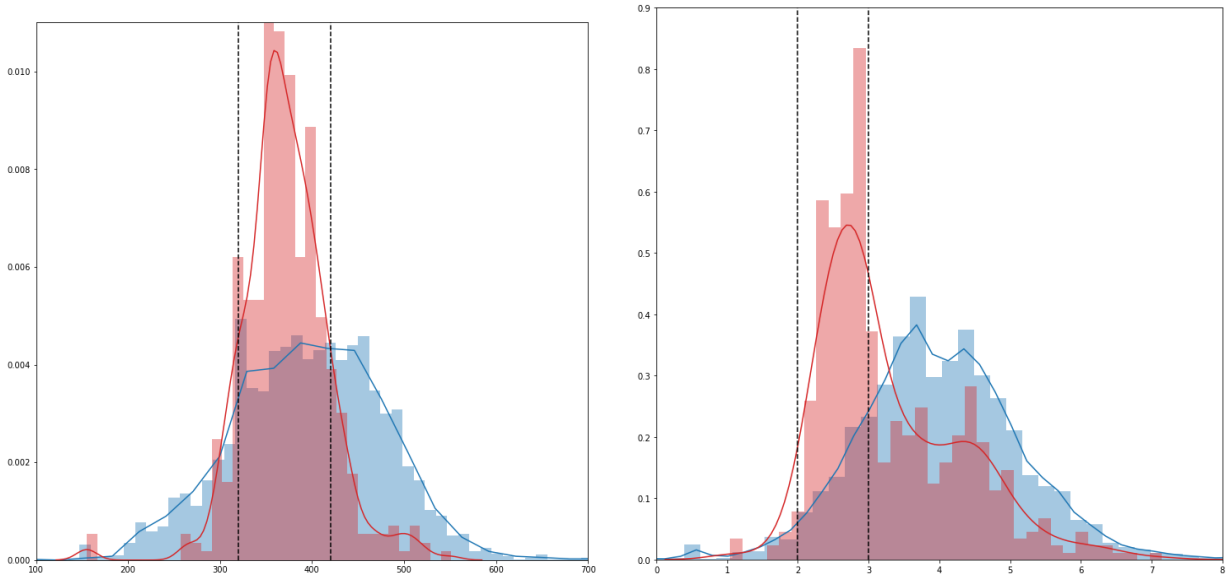


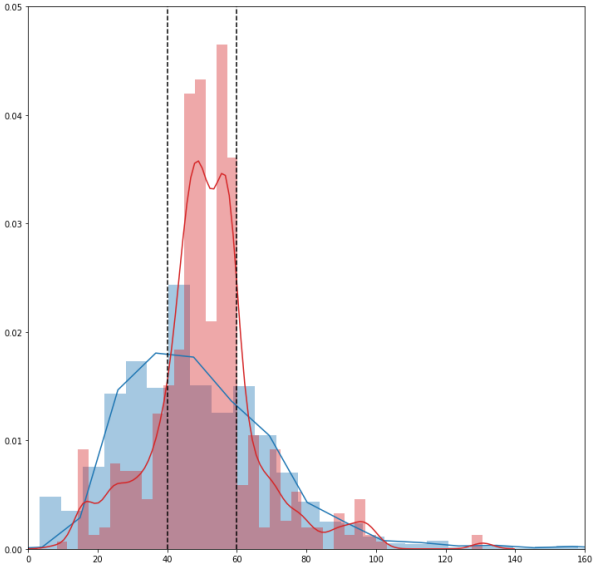
Figure 3. The percentage of valid molecules and the percentage of molecules that fulfil the conditions specified in table 2. The different lines represent, from top to bottom: the validity of the molecule, molecular weight, clogP, PSA and molecules that fulfil all objectives.

Table 3. The percentage of molecules that satisfy each constraint in both the initial set and the set of molecules generated in the last 10 epochs.

Property	Set of lead molecules	AI-generated
Valid molecules	100%	93.2%
Molecular weight	70.5%	72.9%
clogP	10.1%	52.1%
PSA	20.4%	66.1%
"sweet spot"	0%	33.4%



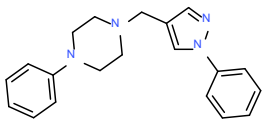
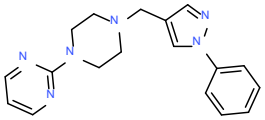
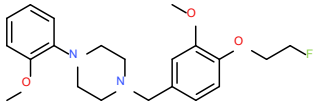
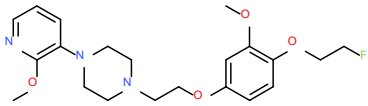
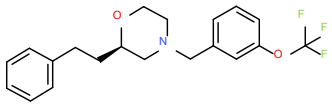
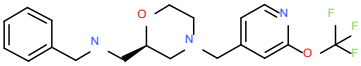
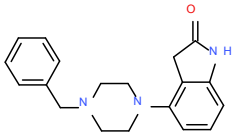
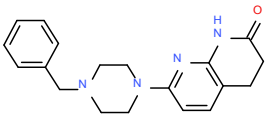
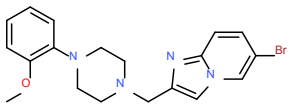
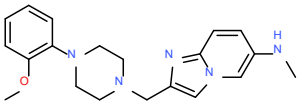
(a) Molecular weight (b) clogP



(c) PSA

Figure 4. The distribution of molecular properties in the original dataset (blue) and in the set of molecules that are generated in the last 10 epochs (red). The targeted ranges, which the aim is to bias the generative model towards, are marked with black dashed lines.

Table 4. Selected pairs of initial lead molecules and their corresponding optimized AI-generated molecule. Targeted properties are color-coded green.

	Initial Lead Compound				AI-Optimized Compound			
No	***Structure***	***MW***	***clogP***	***PSA***	***Structure***	***MW***	***clogP***	***PSA***
1	 CHEMBL210405	318.2	3.2	24		320.2	2.0	50
2	 CHEMBL256492	374.2	3.4	34		405.2	2.7	56
3	 CHEMBL3335542	365.2	4.4	22		381.2	3.0	47
4	 CHEMBL314952	307.2	2.5	36		322.2	2.3	49
5	 CHEMBL211164	400.1	3.4	33		351.2	2.7	45

Discussion

The aim of this paper was primarily to present the concept of how reinforcement learning can be used to generate new compounds while operating on molecular fragments. This approach shows several promising traits and is able to learn how to automatically design molecules that fulfill sweet spot criteria. Specifically, the agent learned that it is, in this case, beneficial to exchange carbon atoms to nitrogen atoms in certain positions (see Table 4). This is a common tactic among medicinal chemists. It has recently been shown that heterocyclic nitrogen's are very frequently introduced when optimizing hit compounds into candidate drugs,[4] and it has proven to be a successful strategy to improve molecular properties.[55] It should be noted that chiral compounds can be generated with this fragment-based approach (Table 4, pair 3), presenting an advantage over some SMILES based generative methods.[17] As is apparent from Table 4, the structural changes between the initial lead and the optimized AI-compound may be seen as minor. This is in line with our approach of focusing on structural similarity as a design strategy. Even though the method was biased to search for molecules similar to the set of input lead molecules, as many as 365 521 unique molecules, out of a 2048 000 generated molecules in total, were created during the learning phase. This shows that the relatively small dopamine D2 and D4 data sets used were sufficiently large for the RL method to learn rules on how fragments could be exchanged to improve targeted properties in a multi-parameter optimization fashion. It is also a testimony to just how vast chemical space is, and that medicinal chemists not always have to look far.[3] While the case study presented here used a rather small dataset of molecules, there is nothing that prevents the method from being used on much larger sets. Using big data can be beneficial, since the RL agent would be able to learn general rules and transformations that improve the properties of the molecules.

The model's freedom of how to replace fragments and the number of allowed replacements is limited to bias the generation to structurally similar compounds. It could, in fact, be beneficial to introduce additional restrictions. For example, in the current case study the agent was allowed to freely exchange any type of fragments in the molecule, as long as they are similar.

Currently, two different similarities, TanimotoMCS and Levenshtein distance of SMILES, are used to measure the similarity of molecules. There are nothing preventing other similarity measures to be used instead of, or in combination, with the current ones. In this context it should be noted that we tested a standard molecular fingerprint (based on the well-known Morgan algorithm), and found it not ideal when comparing fragments. Other types of fingerprints such as the atom-atom-path similarity published by Gobbi *et al.*[56] may be a better alternative.

There are additional features implemented in the presented framework to further control the behavior of the agent. That is, compounds including certain unwanted fragments can be prevented from being generated by treating them in the same way as non-valid SMILES strings. Also, an extra-large reward can be given if the agent includes desirable fragments in the generated molecule. Technically this is achieved by substructure searching using SMARTS matching as implemented in RDKit.[42] Excluding and including certain fragments mimic typical project work, where parts of molecules often remains untouched when working on a "structural series". That is, medicinal chemists tend to retain certain functional groups in their designs for good reasons like maintaining essential ligand-protein interactions and for being able to use established synthetic routes with available chemical intermediates. The piperazine ring in the dopamine D4 compounds would be an example of such a desired fragment to reward inclusion of (Table 4), and a hydrazine moiety could be an example of an unwanted fragment.

The current, standard *de novo* RL generative approaches[16-18] rely on character probability distributions when creating new SMILES strings. This may introduce a risk that the generative model ends up repeating a specific pattern. For example, the underlying distributions of aliphatic carbons in SMILES strings may result in molecules with long strings of carbon atoms. This can, however, not happen with our method, since the model is only allowed to construct new molecules by exchanging similar fragments.

Conclusions

In summary, we have devised and demonstrated a *de novo* method for the automatic generation of molecules with the ultimate goal of improving quality and efficiency in drug discovery. The method uses a reinforcement learning approach that modifies molecules on a fragment level. It is based on an actor-critic model and learns how to modify and improve molecules for them to have desired properties. The presented method is designed to generate molecules which are similar to a given set of lead molecules.

We report an alternative way of representing molecules and their fragments by encoding them as binary strings, with the objective that similar fragments should have similar binary strings. The similarity between fragments is assessed by a combination of a maximum common substructure derived measure and the Levenshtein distance between SMILES strings. The encodings are created with the help of a binary tree, where the most similar fragments are sorted into neighboring leaves. This allows the method to replace fragments in a molecule with similar fragments in a controlled manner.

In a case study, we show that the presented method can handle multi-objective optimization to generate unique drug candidates with desired properties. That is, a third of all AI-generated molecules ended up in the defined sweet spot, fulfilling all targeted properties, while there were none in the initial set of lead compounds.

The system is modular and other criteria and models can easily be hooked up. For example, at AstraZeneca we have included internal predictive models for DMPK and safety (e.g. solubility, permeability, clearance and hERG models), and exploit 3D information (such as predicted binding affinities from docking scores or shape-matching) in the multiparameter optimization of molecules used in real-life drug hunting projects.

In addition to being specifically biased towards generating structurally similar molecules (to existing leads), an advantage over SMILES-based generative *de novo* methods is the ability to generate stereoisomers. The method is user-friendly and does not require any time-consuming training before use. We call the method DeepFMPO.

Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: http. The data sets used, and the full implementation of the conducted experiment is available at https://github.com/stan-his/DeepFMPO

Acknowledgements

The authors thank David A. Cosgrove (CozChemIx) and Peter Brandt (AstraZeneca) for valuable discussions.

References

(1) Bohacek, R. S.; McMartin, C.; Guida, W. C. The art and practice of structure-based drug design: a molecular modeling perspective. *Med. Res. Rev.* **1996**, *16*, 3–50.

(2) Martin, Y. C.; Kofron, J. L.; Traphagen, L. M. Do structurally similar molecules have similar biological activity? *J. Med. Chem.* **2002**, *45*, 4350–4358.

(3) Giordanetto, F.; Boström, J.; Tyrchan, C. Follow-on drugs: How far should chemists look? *Drug Discovery Today* **2011**, *16*, 722–732.

(4) Brown, D. G.; Boström, J. Where do recent small molecule clinical development candidates come from? *J. Med. Chem.* **2018**, *61*, 9442–9468.

(5) Murcko, M. What makes a great medicinal chemist. *J. Med. Chem.* **2018**, *61*, 7419–7424.

(6) Walters W. P.; Stahl M. T.; Murcko M A. Virtual screening – an overview. *Drug Discovery Today* **1998**, *3*, 160–178.

(7) Lyu, J.; Wang, S.; Balius, T. E.; Singh, I.; Levit, A.; Moroz, Y. S.; O'Meara, M. J.; Che, T.; Algaa, E.; Tolmachova, K.; Tolmachev, A. A.; Shoichet, B. K.; Roth B. L.; Irwin J. J. Ultra-large library docking for discovering new chemotypes. *Nature* **2019**, *566*, 224–229.

(8) Hoffmann, T.; Gastreich, M., The next level in chemical space navigation: going far beyond enumerable compound libraries. *Drug Discovery Today* **2019**. *In Press*

(9) FastROCS; OpenEye Scientific Software, Inc.: Santa Fe, NM, U.S.; https://www.eyesopen.com/molecular-modeling-fastrocs.

(10) Melville, J. L.; Burke, E. K.; Hirst, J. D. Machine learning in virtual screening. *Combinatorial chemistry & high throughput screening* **2009**, *12*, 332–343.

(11) Muegge, I.; Oloff, S. Advances in virtual screening. *Drug Discovery Today Technologies* **2006**, *3*, 405–411.

(12) Lavecchia, A. Machine-learning approaches in drug discovery: methods and applications. *Drug Discovery Today* **2015**, *20*, 318–331.

(13) Schneider, G.; Fechner, U. Computer-based de novo design of drug-like molecules, *Nat. Rev. Drug Discov*. **2005**, *4*, 649–663.

(14) Sanchez-Lengeling, B.; Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* **2018**, *361*, 360–365.

(15) Devi, R. V.; Sathya, S. S.; Coumar, M. S. Evolutionary algorithms for de novo drug design–A survey. *Applied Soft Computing* **2015**, *27*, 543–552.

(16) Gupta, A.; Müller, A. T.; Huisman, B. J. H.; Fuchs, J. A.; Schneider, P.; Schneider, G. Generative Recurrent Networks for De Novo Drug Design. *Mol Inform.* **2018**, 37, 1700111.

(17) Olivecrona, M.; Blaschke, T.; Engquist, O.; Chen, H. Molecular de-novo design through deep reinforcement learning. *J. Cheminformatics* **2017**, *9*, 48.

(18) Popova, M.; Isayev, O.; Tropsha, A. Deep reinforcement learning for de novo drug design. *Science Advances* **2018**, *4*, eaap7885.

(19) Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **1992**, *8*, 229–256.

(20) Weininger. D. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comp. Sci.* **1988**, *28*, 31-36.

(21) Grant, A. J.; Haigh, J. A.; Pickup, B. T.; Nicholls, A.; Sayle, R. A. Lingos, finite state machines, and fast similarity searching. *J. Chem. Inf. Model.*, **2006**, *46*, 1912–1918.

(22) Arús-Pous. J.; Blaschke, T.; Ulander, S.; Reymond, J. -L.; Chen H.; Engqvist, O. Exploring the GDB-13 chemical space using deep generative models. *J. Cheminformatics*, **2019**, *11*, 20-34.

(23) Brooks, W. H.; Guida, W.C.; Daniel, K. G.; The significance of chirality in drug design and development. *Curr Top Med Chem.* **2011**,*11*, 760-70.

(24) O'Boyle, N.; Dalke, A. DeepSMILES: An adaptation of SMILES for use in machine-learning of chemical structures. https://doi.org/10.26434/chemrxiv.7097960.v1

(25) Elton, D. C.; Boukouvalas, Z.; Fuge, M. D.; Chung, P. W. Deep learning for molecular generation and optimization – a review of the state of the art. *arXiv e-prints* **2019**, arXiv:1903.04388.

(26) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.,* **2018**, *4*, 268–276.

(27) Kang, S.; Cho, K. Conditional molecular design with deep generative models. *J. Chem Inf. Model*, **2019**, *59*, 43–52.

(28) Kingma, D. P.; Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* **2013**,

(29) Li, Y.; Zhang, L.; Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *J. Cheminformatics* **2018**, *10*, 33-57.

(30) Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. *Advances in Neural Information Processing Systems*. **2014**, 2672–2680.

(31) Metz, L.; Poole, B.; Pfau, D.; Sohl-Dickstein, J. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163* **2016**

(32) Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein generative adversarial networks. International Conference on Machine Learning. **2017**, 214–223.

(33) Skalic, M.; Jiménez, J.; Sabbadin, D.; De Fabritiis, G. Shape-based generative modeling for de-novo drug design, *J. Chem. Inf. Model.*, **2019**, *59*, 1205–1214.

(34) Kogej, T.; Blomberg, N.; Greasley, P. J. Mundt, S.; Vainio, M. J.; Schamberger, J.; Schmidt, G.; Hüser, J. Big pharma screening collections: more of the same or unique libraries? The AstraZeneca–Bayer Pharma AG case. *Drug Discovery Today.* **2013**, *18*, 1014-1024

(35) Pineda, F. J. Generalization of back-propagation to recurrent neural networks. *Phys. Rev. Lett.* **1987**, *59*, 2229–2232.

(36) Schuster, M.; Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681.

(37) Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. *Proc. Int. Conf. Mach. Learn.* Atlanta, Georgia, USA, 2013; 1310–1318.

(38) Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780.

(39) Sutton, R. S.; Barto, A. G. *Reinforcement Learning: An Introduction* MIT press Cambridge, 1998; Vol. 135.

(40) Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine Learning* **1988**, *3*, 9–44.

(41) Kawai, K.; Nagata, N.; Takahashi, Y. De novo design of drug-like molecules by a fragment-based molecular evolutionary approach. *J. Chem. Inf. Model.* **2014**, *54*, 49–56.

(42) Landrum G. RDKit: Open-source cheminformatics. http://www.rdkit.org.

(43) Sheridan, R. P.; Kearsley S. K. Why do we need so many chemical similarity search methods? *Drug Discovery Today.* **2002**, *7*, 903-911.

(44) Boström, J.; Hogner, A.; Schmitt, S. Do structurally similar ligands bind in a similar fashion? *J. Med. Chem.* **2006**, *49*, 6716–6725.

(45) Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*. **1966**, 10, 707–710.

(46) Hann, M. M.; Keserü, G. M. Finding the sweet spot: the role of nature and nurture in medicinal chemistry. *Nat. Rev. Drug Discov.* **2012**, *11*, 355-365.

(47) Leo, A.; Hansch, C.; Elkins, D. Partition coefficients and their uses. *Chem Rev,* **1971**, *71*, 525–616.

(48) Coley, C. W.; Rogers, L.; Green, W. H.; Jensen, K. F. SCScore: Synthetic Complexity Learned from a Reaction Corpus. *J. Chem. Inf. Model.*, **2018**, 58, 252–261.

(49) Dalke, A.; Bache, E.; Van De Waterbeemd, H.; Boyer, S. C-Lab: a web tool for physical property and model calculations; AstraZeneca: Mölndal, Sweden. http://dalkescientific.com/writings/CLab-EuroQSAR2008.pdf

(50) Bento, A. P.; Gaulton, A.; Hersey, A.; Bellis, L. J.; Chambers, J.; Davies, M.; Krüger, F. A.; Light, Y.; Mak, L.; McGlinchey, S.; Nowotka, M.; Papadatos, G.; Santos, R.; Overington, J. P. The ChEMBL bioactivity database: an update. *Nucleic Acids Research,* **2014**, *42*, D1083-D1090. http://doi.org/10.6019/CHEMBL.database.24

(51) Boström, J.; Gundertofte, K.; Liljefors, T. A pharmacophore model for dopamine D4 receptor antagonists", *J. Comput.-Aid. Mol. Des.*, **2000**, *14*, 769-786

(52) Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* **2014**,

(53) Chollet, François. Keras. **2015**, https://keras.io.

(54) Al-Rfou, Rami, et al. "Theano: A Python framework for fast computation of mathematical expressions." *arXiv preprint arXiv:1605.02688* **2016**.

(55) Pennington, L. D.; Moustakas, D. T. The necessary nitrogen atom: a versatile highimpact design element for multiparameter optimization. *J. Med. Chem.* **2017**, *60*, 3552–3579.

(56) Gobbi, A.; Giannetti, A. M.; Chen, H.; Lee, M. -L. Atom-Atom-Path similarity and Sphere Exclusion clustering: tools for prioritizing fragment hits J.Cheminformatics 2015, 7:11