RESEARCH

Molecular De-Novo Design through Deep Reinforcement Learning

Marcus Olivecrona*, Thomas Blaschke, Ola Engkvist and Hongming Chen

Abstract

This work introduces a method to tune a sequence-based generative model for molecular *de novo* design that through augmented episodic likelihood can learn to generate structures with certain specified desirable properties. We demonstrate how this model can execute a range of tasks such as generating analogues to a query structure and generating compounds predicted to be active against a biological target. As a proof of principle, the model is first trained to generate molecules that do not contain sulphur. As a second example, the model is trained to generate analogues to the drug Celecoxib, a technique that could be used for scaffold hopping or library expansion starting from a single molecule. Finally, when tuning the model towards generating compounds predicted to be active against the dopamine receptor D2, the model generates structures of which more than 95% are predicted to be active, including experimentally confirmed actives that have not been included in either the generative model nor the activity prediction model.

Keywords: *De Novo* design; Recurrent Neural Networks; Reinforcement Learning

1 Introduction

Drug discovery is often described using the metaphor of finding a needle in a haystack. In this case, the haystack comprises the on the order of $10^{60} - 10^{100}$ synthetically feasible molecules [1], out of which we need to find an compound which satisfies the plethora of criteria such as bioactivity, drug metabolism and pharmacokinetic (DMPK) profile, synthetic accessibility, etc. The fraction of this space that we can synthesize and test at all - let alone efficiently - is negligible. By using algorithms to virtually design and assess molecules, *de novo* design offers ways to reduce the chemical space into something more manageable for the search of the needle.

Early *de novo* design algorithms [1] used structure based approaches to grow ligands to sterically and electronically fit the binding pocket of the target of interest [2, 3]. A limitation of these methods is that the molecules created often possess poor DMPK properties and can be synthetically intractable. In contrast, the ligand based approach is to by some means generate a large virtual library of chemical structures, and search this chemical space using a scoring function that typically takes into account several properties such as DMPK profiles, synthetic accessibility, bioactivity, and query structure similarity [4, 5]. One way to create such a virtual library is to use known chemical reactions alongside a set of available chemical building blocks, resulting in a large number of synthetically accessible structures [6]; another possibility is to use transformational rules based on the expertise of medicinal chemists to design analogues to a query structure. For example, Besnard et al. [7] applied a transformation rule approach to the design of novel dopamine receptor D2 (DRD2) receptor active compounds with specific polypharmacological profiles and appropriate DMPK profiles for a CNS indication. Although using either transformation or reaction rules can reliably and effectively generate novel structures, they are limited by the inherent rigidness and scope of the predefined rules and reactions.

A third approach, known as inverse Quantitative Structure Activity Relationship (inverse QSAR), tackles the problem from a different angle: rather than first generating a virtual library and then using a QSAR model to score and search this library, inverse QSAR aims to map a favourable region in terms of predicted activity to the corresponding molecular structures [8–10]. This is not a trivial problem: first the solutions of molecular descriptors corresponding to the region need to be resolved using the QSAR model, and these then need be mapped back to the corresponding molecular structures. The fact that the molecular descriptors

*Correspondence: m.olivecrona@gmail.com
External Sciences, Discovery Sciences, Innovative Medicines and Early Development Biotech Unit, AstraZeneca R&D Gothenburg, 43183 Mölndal, Sweden
Full list of author information is available at the end of the article

chosen need to be suitable both for building a forward predictive QSAR model as well as for translation back to molecular structure is one of the major obstacles for this type of approach.

The Recurrent Neural Network (RNN) is commonly used as a generative model for data of sequential nature, and have been used successfully for tasks such as natural language processing [11] and music generation [12]. Recently, there has been an increasing interest in using this type of generative model for the *de novo* design of molecules [13–15]. By using a data-driven method that attempts to learn the underlying probability distribution over a large set of chemical structures, the search over the chemical space can be reduced to only molecules seen as reasonable, without introducing the rigidity of rule based approaches. Segler *et. al.* demonstrated that an RNN trained on the canonicalized SMILES representation of molecules can learn both the syntax of the language as well as the distribution in chemical space [13]. They also show how further training of the model using a focused set of actives towards a biological target can produce a fine-tuned model which generates a high fraction of predicted actives.

In two recent studies, reinforcement learning [16] was used to fine tune pre-trained RNNs. Yu *et. al.* [15] use an adversarial network to estimate the expected return for state-action pairs sampled from the RNN, and by increasing the likelihood of highly rated pairs improves the generative network for tasks such as poem generation. Jaques *et. al.* [17] use Deep Q-learning to improve a pre-trained generative RNN by introducing two ways to score the sequences generated: one is a measure of how well the sequences adhere to music theory, and one is the likelihood of sequences according to the initial pre-trained RNN. Using this concept of prior likelihood they reduce the risk of forgetting what was initially learnt by the RNN, compared to a reward based only on the adherence to music theory. The authors demonstrate significant improvements over both the initial RNN as well as a reinforcement learning only approach. They later extend this method to several other tasks including the generation of SMILES, and optimize toward molecular properties such as cLogP [18] and QED drug-likeness [19]. However, they report that the method can lead to exploitation of the reward resulting in unrealistically simple molecules that are more likely to satisfy the optimization requirements than more complex structures [17].

In this study we propose a, to our knowledge, novel approach using reinforcement learning to tune RNNs for episodic tasks [16], in this case the task of generating molecules with given desirable properties. Through learning an augmented episodic likelihood which is a composite of prior likelihood [17] and a user defined scoring function, the method aims to fine-tune an RNN pre-trained on the ChEMBL database [20] towards generating desirable compounds. Compared to maximum likelihood estimation finetuning [13], this method can make use of negative as well as continuous scores, and may reduce the risk of catastrophic forgetting [21]. The method is applied to several different tasks of molecular *de novo* design, and an investigation was carried out to illustrate how the method affects the behaviour of the generative model on a mechanistic level.

2 Methods

2.1 Recurrent Neural Networks

A recurrent neural network is an architecture of neural networks designed to make use of the symmetry across steps in sequential data while simultaneously at every step keep track of the most salient information of previously seen steps, which may affect the interpretation of the current one [22]. It does so by introducing the concept of a *cell* (Figure 1). For any given step t, the $cell_t$ is a result of the previous $cell_{t-1}$ and the current input x^{t-1}. The content of $cell_t$ will determine both the output at the current step as well as influence the next cell state. The cell thus enables the network to have a memory of past events, which can be used when deciding how to interpret new data. These properties make an RNN particularly well suited for problems in the domain of natural language processing. In this setting, a sequence of words can be encoded into one-hot vectors the length of our vocabulary X. Two additional tokens, GO and EOS, may be added to denote the beginning and end of the sequence respectively.

2.1.1 Learning the data

Training an RNN to learn from sequential data is typically done by maximum likelihood estimation, i.e. by maximizing the predicted log likelihood of the next token x^t in the target sequence given tokens for the previous steps (Figure 1). At every step the model will produce a probability distribution over what the next character is likely to be, and the aim is to maximize the likelihood assigned to the correct token:

$$L(\Theta) = -\sum_{t=1}^{T} \log P(x^t \mid x^{t-1}, ..., x^1)$$

The cost function $L(\Theta)$, often applied to a subset of all training examples known as a batch, is minimized with respect to the network parameters Θ. Given a predicted log likelihood $\log P$ of the target at step t, the gradient of the prediction with respect to Θ is used to make an update of Θ. This method of fitting a neural network

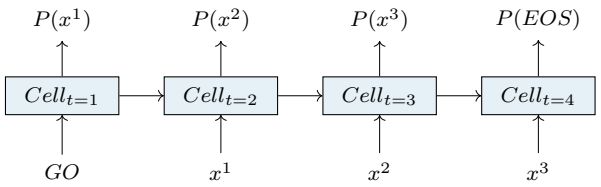

Figure 1 Learning the data. Depiction of maximum likelihood training of an RNN. x^t are the target sequence tokens we are trying to learn by maximizing $P(x^t)$ for each step.

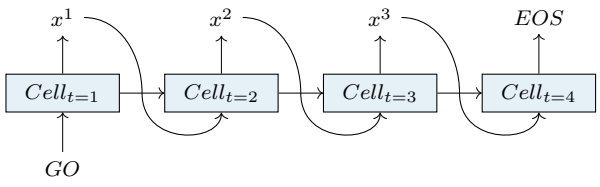

Figure 2 Generating sequences. Sequence generation by a trained RNN. Every timestep t we sample the next token of the sequence x^t from the probability distribution given by the RNN, which is then fed in as the next input.

is called back-propagation. Due to the architecture of the RNN, changing the network parameters will not only affect the direct output at time t, but also affect the flow of information from the previous cell into the current one iteratively. This domino-like effect that the recurrence has on back-propagation gives rise to some particular problems, and back-propagation applied to RNNs is referred to as back-propagation through time (BPTT).

BPTT is dealing with gradients that through the chain-rule contains terms which are multiplied by themselves many times, and this can lead to a phenomenon known as exploding and vanishing gradients. If these terms are not unity, the gradients quickly become either very large or very small. In order to combat this issue, Hochreiter et al. introduced the Long-Short-Term Memory cell [23], which through a more controlled flow of information can decide what information to keep and what to discard. The Gated Recurrent Unit is a simplified implementation of the Long-Short-Term Memory architecture that achieves much of the same effect at a reduced computational cost [24].

2.1.2 Generating new samples

Once an RNN has been trained on target sequences, it can then be used to generate new sequences that follow the conditional probability distributions learned from the training set. The first input - the *GO* token - is given and at every timestep after we sample an output token x^t from the predicted probability distribution $P(X^t)$ over our vocabulary X and use x^t as our next input. Once the *EOS* token is sampled, the sequence is considered finished (Figure 2).

2.1.3 Tokenizing and one-hot encoding SMILES

A SMILES [25] string represents a molecule as a sequence of characters corresponding to atoms as well as special characters denoting opening and closure of rings and branches. The SMILES string is, in most cases, tokenized based on single character, except for atom types which comprise two characters such as "Cl" and "Br" and special environments denoted by square brackets (e.g [nH]), where they are considered as one token. This method of tokenization resulted in 86 tokens present in the training data (see Appendix for the full list). Figure 3 exemplifies how a chemical structure is translated to both the SMILES and one-hot encoded representations.

There are many different ways to represent a single molecule using SMILES. Algorithms that always represent a certain molecule with the same SMILES are referred to as canonical. However, different implementations of the algorithms can still produce different SMILES.

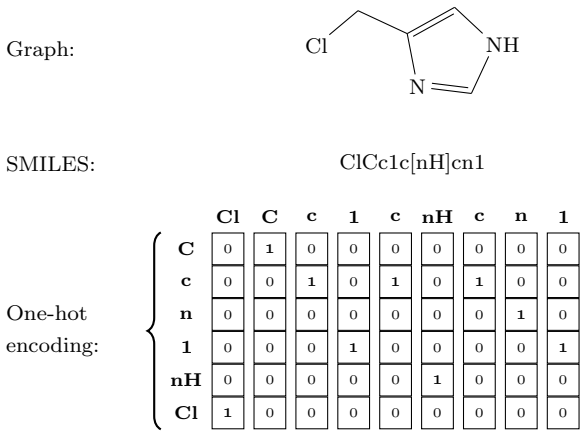

Figure 3 Three representations of 4-(chloromethyl)-1H-imidazole. Depiction of a one-hot representation derived from the SMILES string of a molecule. Here a reduced vocabulary is shown, while in practice a much larger vocabulary that covers all tokens present in the training data is used.

2.2 Reinforcement Learning

Consider an Agent, that given a certain state $s \in \mathbb{S}$ has to choose which action $a \in \mathbb{A}(s)$ to take, where $\mathbb{S}$ is the set of possible states and $\mathbb{A}(s)$ is the set of possible

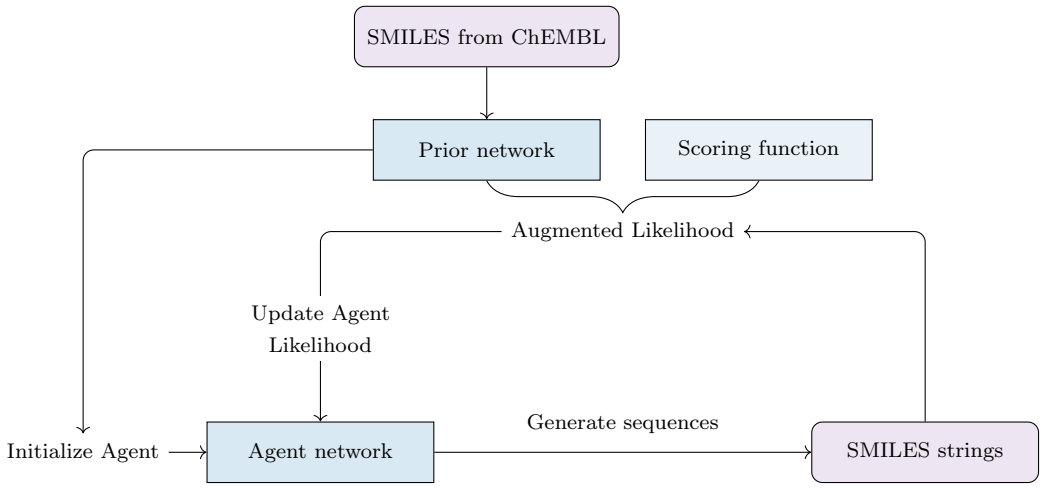

Figure 4 The Agent. Illustration of how the model is constructed. Starting from a Prior network trained on ChEMBL, the Agent is trained using the augmented likelihood of the SMILES generated.

actions for that state. The policy $\pi(a \mid s)$ of an Agent maps a state to the probability of each action taken therein. Many problems in reinforcement learning are framed as Markov decision processes, which means that the current state contains all information necessary to guide our choice of action, and that nothing more is gained by also knowing the history of past states. For most real problems, this is an approximation rather than a truth; however, we can generalize this concept to that of a partially observable Markov decision process, in which the Agent can interact with an incomplete representation of the environment. Let $r(a \mid s)$ be the reward which acts as a measurement of how good it was to take an action at a certain state, and the long-term return $G = \sum_{t=1}^{T} r_t$ as the cumulative rewards collected up to time T. Since molecular desirability in general is only sensible for a completed SMILES, we will refer only to the return of a complete sequence.

What reinforcement learning concerns, given a set of actions taken from some states and the rewards thus received, is how to improve the Agent policy in such a way as to increase the expected return $\mathbb{E}[G]$. A task which has a clear endpoint at step T is referred to as an episodic task [16], where T corresponds to the length of the episode. Generating a SMILES is an example of an episodic task, which ends once the *EOS* token is sampled.

The states and actions used to train the agent can be generated both by the agent itself or by some other means. If they are generated by the agent itself the learning is referred to as *on-policy*, and if they are generated by some other means the learning is referred to as *off-policy* [16].

2.3 The Prior network

Maximum likelihood estimation was employed to train the initial RNN composed of 3 layers with 1024 Gated Recurrent Units (forget bias 5) in each layer. The RNN was trained on the RDKit [26] canonical SMILES of 1.5 million structures from ChEMBL [20] where the molecules were restrained to containing between 10 and 50 heavy atoms and elements $\in \{H, B, C, N, O, F, Si, P, S, Cl, Br, I\}$. The model was trained with stochastic gradient descent for 50 000 steps using a batch size of 128, utilizing the Adam optimizer [27] ($\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$) with an initial learning rate of 0.001 and a 0.02 learning rate decay every 100 steps. Gradients were clipped to $[-3, 3]$. Tensorflow [28] was used to implement the Prior as well as the reinforcement learning Agent.

2.4 The Agent network

We now frame the problem of generating a SMILES representation of a molecule with specified desirable properties via an RNN as a partially observable Markov decision process, where the agent must make a decision of what character to choose next given the current cell state. We use the probability distributions learnt by the aforementioned prior model as our initial prior policy. We will refer to the network using the prior policy simply as the *Prior*, and the network whose policy has since been modified as the *Agent*. The task is episodic, starting with the first step of the RNN and ending when the *EOS* token is sampled. The sequence of actions $A = a_1, a_2, ..., a_T$ during this episode represents the SMILES generated and the product of the action probabilities $P(A) = \prod_{t=1}^{T} \pi(a_t \mid s_t)$ represents the model likelihood of the sequence formed. Let $S(A) \in [-1, 1]$

be a scoring function that rates the desirability of the sequences formed using some arbitrary method. The goal now is to update the agent policy π from the prior policy π_{prior} in such a way as to increase the expected score for the generated sequences. However, we do want our new policy to be anchored to the prior policy, which we presume has learnt something about both the syntax of SMILES and distribution of molecular structure in ChEMBL. We therefore denote an augmented likelihood $\log P(A)_{\mathbb{U}}$ as a prior likelihood modulated by the desirability of a sequence:

$$\log P(A)_{\mathbb{U}} = \log P(A)_{prior} + \sigma S(A)$$

where σ is a scalar coefficient. The return $G(A)$ of a sequence A can in this case be seen as the agreement between the Agent likelihood $\log P(A)_{\mathbb{A}}$ and the augmented likelihood:

$$G(A) = -[\log P(A)_{\mathbb{U}} - \log P(A)_{\mathbb{A}}]^2$$

The goal of the Agent is to learn a policy which maximizes the expected return, achieved by minimizing the cost function $L(\Theta) = -G$. The Agent is trained in an on-policy fashion on batches of 128 generated sequences, making an update to π after every batch has been generated and scored. A standard gradient descent optimizer with a learning rate of 0.0005 was used and gradients were clipped to $[-3, 3]$.

Figure 4 shows an illustration of how the Agent, initially identical to the Prior, is trained using reinforcement learning. The training shifts the probability distribution from that of the Prior towards a distribution modulated by the desirability of the structures.

2.5 The DRD2 activity model

In one of our studies the objective of the Agent is to generate molecules that are predicted to be active against a biological target. The dopamine type 2 receptor DRD2 was chosen as the target, and corresponding bioactivity data was extracted from ExCAPE-DB [29]. In this dataset there are 7218 actives (pIC50 > 5) and 343204 inactives (pIC50 < 5). A subset of 100 000 inactive compounds was randomly selected. In order to decrease the nearest neighbour similarity between the training and testing structures, the actives were grouped in clusters based on their molecular similarity. The Jaccard [30] index, for binary vectors also known as the Tanimoto similarity, based on the RDKit implementation of binary Extended Connectivity Molecular Fingerprints with a diameter of 6 (ECFP6 [31]) was used as a similarity measure and the actives were clustered using the Butina clustering algorithm [32] in RDKit with a clustering cutoff of 0.4. In this algorithm, centroid molecules will be selected, and everything with a similarity higher than 0.4 to these centroids will be assigned to the same cluster. The centroids are chosen such as to maximize the number of molecules that are assigned to any cluster. The clusters were sorted by size and iteratively assigned to test, validation, and train (assigned 4 clusters each iteration) to give a distribution of $\frac{1}{6}$, $\frac{1}{6}$, and $\frac{4}{6}$ of the clusters respectively. The inactive compounds, of which less than 0.5% were found to belong to any of the clusters formed by the actives, were split randomly into the three sets using the same ratios.

A support vector machine (SVM) classifier with a Gaussian kernel was built in Sci-Kit Learn [33] on the training set as a predictive model for DRD2 activity. The optimal C and Gamma values utilized in the final model were obtained from a grid search for the highest ROC-AUC performance on the validation set.

3 Results and Discussion

3.1 Structure generation by the Prior

After the initial training, 94% of the sequences generated by the Prior as described in Section 2.1.2 corresponded to valid molecular structures according to RDKit [26] parsing, out of which 90% are novel structures outside of the training set. A set of randomly chosen structures generated by the Prior, as well as by Agents trained in the subsequent examples, are shown in the Appendix. The process of generating a SMILES by the Prior is illustrated in Figure 5. For every token in the generated SMILES sequence, the conditional probability distribution over the vocabulary at this step according to the Prior is displayed. The sequence of distributions are depicted in Figure 5. For the first step, when no information other than the initial GO token is present, the distribution is an approximation of the distribution of first tokens for the SMILES in the ChEMBL training set. In this case "O" was sampled, but "C", "N", and the halogens were all likely as well. Corresponding log likelihoods were -0.3 for "C", -2.7 for "N", -1.8 for "O", and -5.0 for "F" and "Cl".

A few (unsurprising) observations:

- Once the aromatic "n" has been sampled, the model has come to expect a ring opening (i.e. a number), since aromatic moieties by definition are cyclic.
- Once an aromatic ring has been opened, the aromatic atoms "c", "n", "o", and "s" become probable, until 5 or 6 steps later when the model thinks it is time to close the ring.
- The model has learnt the RDKit SMILES format of increasing ring numbers, and expects the first ring to be numbered "1". Ring numbers can be reused, as in the two first rings in this example.

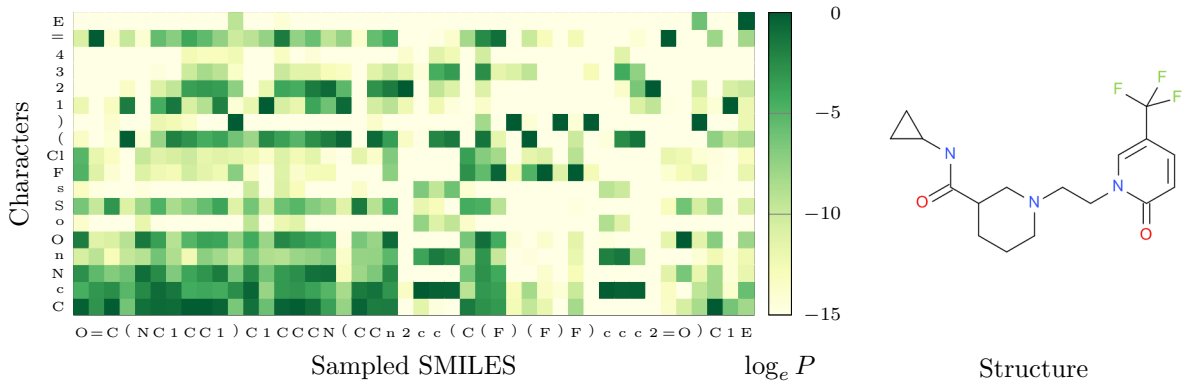

Figure 5 How the model thinks while generating the molecule on the right. Conditional probability over the next token as a function of previously chosen ones according to the model. On the y-axis is shown the probability distribution for the character to be choosen at the current step, and on the x-axis is shown the character that in this instance was sampled. E = EOS.

Only once "1" has been sampled does it expect a ring to be numbered "2" and so on.

3.2 Learning to avoid sulphur

As a proof of principle the Agent was first trained to generate molecules which do not contain sulphur, by defining the scoring function as:

$$S(A) = \begin{cases} 1 & \text{if valid and no S} \\ 0 & \text{if not valid} \\ -1 & \text{if contains S} \end{cases}$$

The Agent was trained for 1000 steps using $\sigma = 2$, and 12800 SMILES sequences were sampled from both the Prior and Agent networks for comparison. After training, only 2% of the structures generated by the Agent contained sulphur, compared to 32% for those generated by the Prior. To explore what effect the training has on the structures generated, relevant properties for non sulphur containing structures generated by both the Prior and the Agent were compared. The molecular weight, cLogP, the number of rotatable bonds, and the number of aromatic rings were all calculated using RDKit. The experiment was repeated three times with different random seeds. There was no considerable difference in the mean values for the two models, however a small increase in the variance was observed for the agent. The results indicate that for sufficiently low learning rate and σ, the structures generated by the Agent adhere to the underlying distribution learnt by the Prior, while achieving a much higher fraction of structures that do not contain sulfur.

Table 1 Comparison of model performance and properties for non-sulphur containing structures generated by the two models. Properties reported as Mean ± StdDev.

Model	Prior	Agent
Fraction of valid SMILES	0.94	0.96
Fraction without sulphur	0.68	**0.98**
Average molecular weight	371 ± 1.70	367 ± 3.30
Average cLogP	3.36 ± 0.04	3.37 ± 0.09
Average NumRotBonds	5.39 ± 0.04	5.41 ± 0.07
Average NumAromRings	2.26 ± 0.02	2.26 ± 0.02

3.3 Similarity guided structure generation

The second task investigated was that of generating structures similar to a query structure. The Jaccard index [30] $J_{i,j}$ of the RDKit implementation of FCFP4 [31] fingerprints was used as a similarity measure between molecules i and j. Compared to the DRD2 activity model (Section 2.5), the feature invariant version of the fingerprints and the smaller diameter 4 was used in order to get a more fuzzy similarity measure. The scoring function was defined as:

$$S(A) = -1 + 2 \times \frac{\min\{J_{i,j}, k\}}{k}$$

This definition means that an increase in similarity is only rewarded up to the point of $k \in [0, 1]$, as well as scaling the reward from -1 (no overlap in the fingerprints between query and generated structure) to 1 (at least k degree of overlap). Celecoxib was chosen as our query structure, and we first investigated whether Celecoxib itself could be generated by using the high values of $k = 1$ and $\sigma = 15$. The Agent was trained for 1000 steps. After a 100 training steps the Agent starts to generate Celecoxib, and after 200 steps it predominately generates this structure (Figure 6).

Celecoxib itself as well as many other similar structures appear in the ChEMBL training set used to build the Prior. An interesting question is whether the Agent could succeed in generating Celecoxib when these struc-

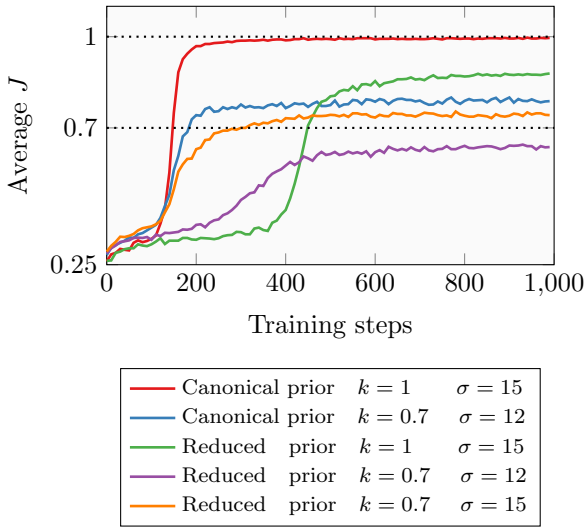

Figure 6 Average similarity J of generated structures as a function of training steps. Difference in learning dynamics for the Agents based on the canonical Prior, and those based on a reduced Prior where everything more similar than $J = 0.5$ to Celecoxib have been removed.

tures are not part of the chemical space covered by the Prior. To investigate this, all structures with a similarity to Celecoxib higher than 0.5 (corresponding to 1804 molecules) were removed from the training set and a new reduced Prior was trained. The prior likelihood of Celecoxib for the canonical and reduced Priors was compared, as well as the ability of the models to generate structures similar to Celecoxib. As expected, the prior probability of Celecoxib decreased when similar compounds were removed from the training set from $\log_e P = -12.7$ to $\log_e P = -19.2$, representing a reduction in likelihood of a factor of 700. An Agent was then trained using the same hyperparameters as before, but on the reduced Prior. After 400 steps, the Agent again managed to find Celecoxib, albeit requiring more time to do so. After 1000 steps, Celecoxib was the most commonly generated structure (about a third of the generated structures), followed by demethylated Celecoxib (also a third) whose SMILES is more likely according to the Prior with $\log_e P = -15.2$ but has a lower similarity ($J = 0.87$), resulting in an augmented likelihood equal to that of Celecoxib.

These experiments demonstrate that the Agent can be optimized using fingerprint based Jaccard similarity as the objective, but making copies of the query structure is hardly useful. A more useful example is that of generating analogues to the query structure, which could be a confirmed active against a biological target found in a High-Throughput-Screen for example. The Agent was therefore trained for 3000 steps, starting from both the canonical as well as the reduced Prior, using $k = 0.7$ and $\sigma = 12$. The Agents based on the canonical Prior quickly converge to their targets, while the Agents based on the reduced Prior converged more slowly. For the Agent based on the reduced Prior where $k = 1$, the fact that Celecoxib and demethylated Celecoxib are given similar augmented likelihoods means that the average similarity converges to around 0.9 rather than 1.0. For the Agent based on the reduced Prior where $k = 0.7$, the lower prior likelihood of compounds similar to Celecoxib translates to a lower augmented likelihood, which lowers the average similarity of the structures generated by the Agent.

To explore whether this reduced prior likelihood could be offset with a higher value of σ, an Agent starting from the reduced Prior was trained using $\sigma = 15$. Though taking slightly more time to converge than the Agent based on the canonical Prior, this Agent too could converge to the target similarity. The learning curves for the different model is shown in Figure 6.

An illustration of how the type of structures generated by the Agent evolves during training is shown in Figure 7. For the Agent based on the reduced Prior with $k = 0.7$ and $\sigma = 15$, three structures were randomly sampled every 100 training steps from step 0 up to step 400. At first, the structures are not similar to Celecoxib. After 200 steps, some features from Celecoxib have started to emerge, and after 300 steps the model generates mostly close analogues of Celecoxib.

We have investigated how various factors affect the learning behaviour of the Agent. In real drug discovery applications, we might be more interested in finding structures with modest similarity to our query molecules rather than very close analogues. For example, one of the structures sampled after 200 steps shown in Figure 7 displays a type of scaffold hopping where the sulphur functional group on one of the outer aromatic rings has been fused to the central pyrazole. The similarity to Celecoxib of this structure is 0.4, which may be a more interesting solution for scaffold-hopping purposes. One can choose hyperparameters and similarity criterion tailored to the desired output. Other types of similarity measures such as pharmacophoric fingerprints [34], Tversky substructure similarity [35], or presence/absence of certain pharmacophores could also be explored.

3.4 Target activity guided structure generation

The third example, perhaps the one most interesting and relevant for drug discovery, is to optimize the Agent towards generating structures with predicted biological activity. This can be seen as a form of inverse QSAR, where the Agent is used to implicitly map high predicted probability of activity to molecular structure.

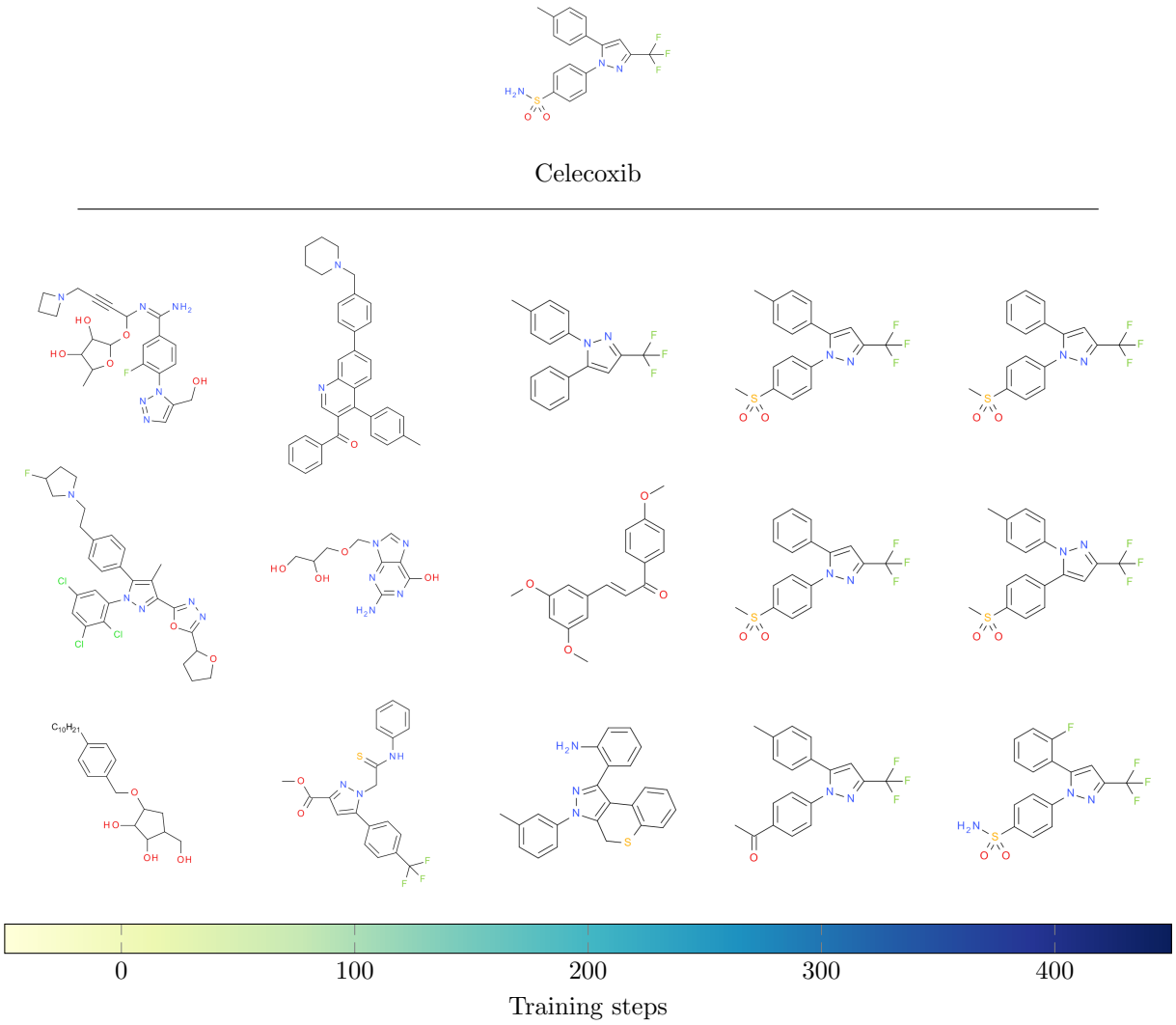

Figure 7 Evolution of generated structures during training Structures sampled every 100 training steps during the training of the Agent towards similarity to Celecoxib with $k = 0.7$ and $\sigma = 15$.

DRD2 was chosen as the biological target. The clustering split of the DRD2 activity dataset as described in Section 2.5 resulted in 1405, 1287, and 4526 actives in the test, validation, and training sets respectively. The average distance to the nearest neighbour in the training set for the test set compounds was 0.53. For a random split of actives in sets of the same sizes this distance was 0.69, indicating that the clustering had significantly decreased training-test set similarity which mimics the hit finding practice in drug discovery to identify diverse hits to the training set. Most of the DRD2 actives are also included in the ChEMBL dataset used to train the Prior. To explore the effect of not having the known actives included in the Prior, a reduced Prior was trained on a reduced subset of the ChEMBL trainingset where all DRD2 actives had been removed.

Table 2 Performance of the DRD2 activity model

Set	Training	Validation	Test
Accuracy	1.00	0.98	0.98
ROC-AUC	1.00	0.99	1.00
Precision	1.00	0.96	0.97
Recall	1.00	0.73	0.82

The optimal hyperparameters found for the SVM activity model were $C = 2^7, \gamma = 2^{-6}$, resulting in a model whose performance is shown in Table 2. The good performance in general can be explained by the apparent difference between actives and inactive compounds as seen during the clustering, and the better

Table 3 Comparison of properties for structures generated by the canonical Prior, the reduced Prior, and corresponding Agents.

Model	Prior	Agent	Prior†	Agent†
Fraction predicted actives	0.03	0.97	0.02	0.96
Fraction similar to train active	0.02	0.79	0.02	0.75
Fraction similar to test active	0.01	0.46	0.01	0.38
Fraction of test actives recovered	0.00	0.13	0.00	0.07
Probability of generating a test set active ($\times 10^{-3}$)	0.17	40.2	0.05	15.0

†DRD2 actives witheld from the training of the Prior

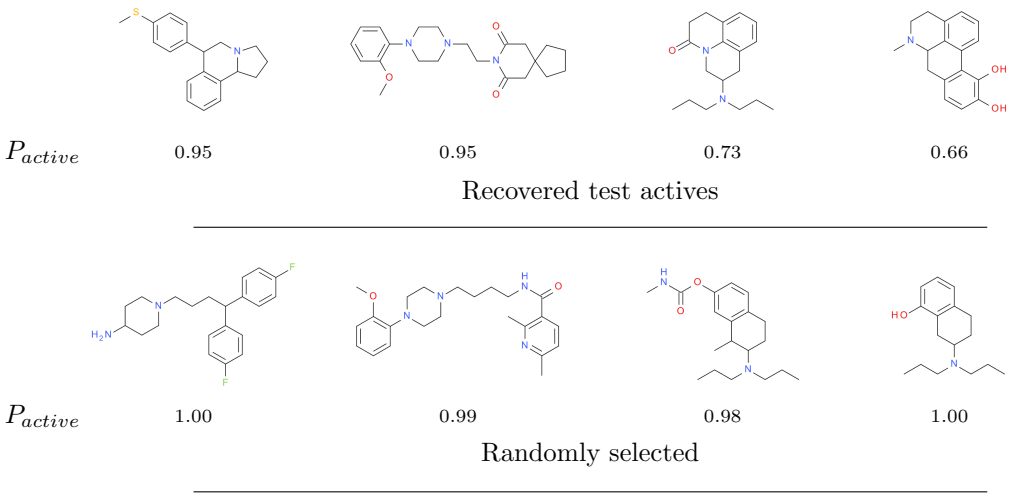

Figure 8 Structures designed by the Agent to target DRD2. Molecules generated by the Agent based on the reduced Prior. On the top are four of the test set actives that were recovered, and below are four randomly selected structures. The structures are annotated with the predicted probability of being active.

performance on the test set compared to the validation set could be due to slightly higher nearest neighbour similarity to the training actives (0.53 for test actives and 0.48 for validation actives).

The output of the DRD2 model for a given structure is an uncalibrated predicted probability of being active P_{active}. This value is used to formulate the following scoring function:

$$S(A) = -1 + 2 \times P_{active}$$

The Agents were trained for 3000 steps using $\sigma = 7$, and 128000 sequences were generated by both the Priors and their corresponding Agents. After training, the fraction of predicted actives according to the DRD2 model increased from 0.02 for structures generated by the reduced Prior to 0.96 for structures generated by the corresponding Agent network (Table 3). To see how well the structure-activity-relationship learnt by the activity model is transferred to the type of structures generated by the Agent RNN, the fraction of compounds with an ECFP6 Jaccard similarity greater than 0.4 to any active in the training and test sets was calculated.

In some cases, the model recovered exact matches from the training and test sets (c.f. Segler *et. al.* [13]). The fraction of recovered test actives recovered by both Priors was 0%. The Agent derived from the canonical Prior managed to recover 13% test actives; the Agent derived from the reduced Prior recovered 7%. For the Agent derived from the reduced Prior, where the DRD2 actives were excluded from the Prior training set, this means that the model has learnt to generate "novel" structures that have been seen neither by the DRD2 activity model nor the Prior, and are experimentally confirmed actives. We can formalize this observation by calculating the probability of a given generated sequence belonging to the set of test actives. For the canonical and reduced Priors, this probability was 0.17×10^{-3} and 0.05×10^{-3} respectively. Removing the actives from the Prior thus resulted in a threefold reduction in the probability of generating a structure from the set of test actives. For the Agents, the probabilities rose to 15.0×10^{-3} and 40.2×10^{-3} respectively, corresponding to an enrichment of a factor of 250 over the Prior models. Again the consequence of removing the actives from the Prior was a threefold reduction in the probability of generating a test set active: the

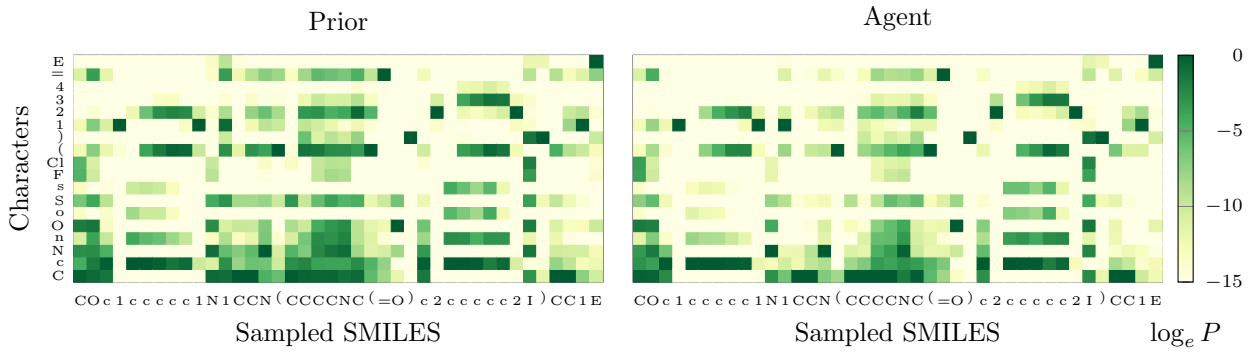

Figure 9 A (small) change of mind. The conditional probability distributions when the DRD2 test set active 'COc1ccccc1N1CCN(CCCCNC(=O)c2ccccc2I)CC1' is generated by the Prior and an Agent trained using the DRD2 activity model, for the case where all actives used to build the activity model have been removed from the Prior. E = EOS.

difference between the two Priors is directly mirrored by their corresponding Agents.

A few of the test set actives generated by the Agent based on the reduced Prior along with a few randomly selected generated structures are shown together with their predicted probability of activity in Figure 8. Encouragingly, the recovered test set actives vary considerably in their structure, which would not have been the case had the Agent converged to generating only a certain type of very similar predicted active compounds.

Removing the known actives from the training set of the Prior resulted in an Agent which shows a decrease in all metrics measuring the overlap between the known actives and the structures generated, compared to the Agent derived from the canonical Prior. Interestingly, the fraction of predicted actives did not change significantly. This indicates that the Agent derived from the reduced Prior has managed to find a similar chemical space to that of the canonical Agent, with structures that are equally likely to be predicted as active, but are less similar to the known actives. Whether or not these compounds are active will be dependent on the accuracy of the target activity model. Ideally, any predictive model to be used in conjunction with the generative model should cover a broad chemical space within its domain of applicability, since it initially has to assess representative structures of the dataset used to build the Prior [13].

Figure 9 shows a comparison of the conditional probability distributions for the reduced Prior and its corresponding Agent when a molecule from the set of test actives is generated. It can be seen that the changes are not drastic with most of the trends learnt by the Prior being carried over to the Agent. However, a big change in the probability distribution even only at one step can have a large impact on the likelihood of the sequence and could significantly alter the type of structures generated.

4 Conclusion

To summarize, we believe that an RNN operating on the SMILES representation of molecules is a promising method for molecular *de novo* design. It is a data-driven generative model that does not rely on pre-defined building blocks and rules, which makes it clearly differentiated from traditional methods. In this study we extend upon previous work [13–15, 17] by introducing a reinforcement learning method which can be used to tune the RNN to generate structures with certain desirable properties through augmented episodic likelihood.

The model was tested on the task of generating sulphur free molecules as a proof of principle, indicating that the agent can find solutions within the space of the Prior while still reflecting the underlying probability distribution initially learnt. To evaluate if the model could be used to generate analogues to a query structure, the Agent was trained to generate structures similar to the drug Celecoxib. Even when all analogues of Celecoxib were removed from the Prior, the Agent could still locate the intended region of chemical space which was not part of the Prior. Further more, when trained towards generating predicted actives against the dopamine receptor DRD2, the Agent generates structures of which more than 95% are predicted to be active, and could recover test set actives even in the case where they were not included in either the activity model nor the Prior. Our results indicate that the method could be a useful tool for drug discovery.

It is clear that the qualities of the Prior are reflected in the corresponding Agents it produces. An exhaustive

study which explores how parameters such as training set size, model size, regularization [36, 37], and training time would influence the quality and variety of structures generated by the Prior would be interesting. Another interesting avenue for future research might be that of token embeddings [38]. The method was found to be robust with respect to the hyperparameters σ and the learning rate.

All of the aforementioned examples used single parameter based scoring functions. In a typical drug discovery project, multiple parameters such as target activity, DMPK profile, synthetic accessibility etc. all need to be taken into account simultaneously. Applying this type of multi-parametric scoring functions to the model is an area requiring further research.

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

List of abbreviations
- DMPK - Drug metabolism and pharmacokinetics
- DRD2 - Dopamine receptor D2
- QSAR - Quantitive structure activity relationship
- RNN - Recurrent neural network
- Log - Natural logarithm
- BPTT - Back-propagation through time
- A - Sequence of tokens constituting a SMILES
- Prior - An RNN trained on SMILES from ChEMBL used as a starting point for the Agent
- Agent - An RNN derived from a Prior, trained using reinforcement learning
- J - Jaccard index
- ECFP6 - Extended Molecular Fingerprints with diameter 6
- SVM - Support Vector Machine
- FCFP4 - Extended Molecular Fingerprints with diameter 4 and feature invariants

Competing interests
The authors declare that they have no competing interests.

Funding
MO, HC, and OE are employed by AstraZeneca. TB has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 676434, "Big Data in Chemistry" ("BIGCHEM", http://bigchem.eu). The article reflects only the authors' view and neither the European Commission nor the Research Executive Agency (REA) are responsible for any use that may be made of the information it contains.

Author's contributions
MO contributed concept and implementation. All authors co-designed experiments. All authors contributed to the interpretation of results. MO wrote the manuscript. HC, TB, and OE reviewed and edited the manuscript. All authors read and approved the final manuscript.

Acknowledgements
The authors thank Thierry Kogej and Christian Tyrchan for general assistance and discussion, and Dominik Peters for his expertise in LaTeX.

References
1. Schneider, G., Fechner, U.: Computer-based de novo design of drug-like molecules. Nat Rev Drug Discov **4**(8), 649–663 (2005). doi:10.1038/nrd1799
2. Böhm, H.-J.: The computer program ludi: A new method for the de novo design of enzyme inhibitors. Journal of Computer-Aided Molecular Design **6**(1), 61–78 (1992). doi:10.1007/BF00124387
3. Gillet, V.J., Newell, W., Mata, P., Myatt, G., Sike, S., Zsoldos, Z., Johnson, A.P.: Sprout: Recent developments in the de novo design of molecules. Journal of Chemical Information and Computer Sciences **34**(1), 207–217 (1994). doi:10.1021/ci00017a027
4. Ruddigkeit, L., Blum, L.C., Reymond, J.-L.: Visualization and virtual screening of the chemical universe database gdb-17. Journal of Chemical Information and Modeling **53**(1), 56–65 (2013). doi:10.1021/ci300535x
5. Hartenfeller, M., Zettl, H., Walter, M., Rupp, M., Reisen, F., Proschak, E., Weggen, S., Stark, H., Schneider, G.: Dogs: Reaction-driven de novo design of bioactive compounds. PLOS Computational Biology **8**, 1–12 (2012). doi:10.1371/journal.pcbi.1002380
6. Schneider, G., Geppert, T., Hartenfeller, M., Reisen, F., Klenner, A., Reutlinger, M., Hähnke, V., Hiss, J.A., Zettl, H., Keppner, S., Spänkuch, B., Schneider, P.: Reaction-driven de novo design, synthesis and testing of potential type ii kinase inhibitors. Future Medicinal Chemistry **3**(4), 415–424 (2011). doi:10.4155/fmc.11.8
7. Besnard, J., Ruda, G.F., Setola, V., Abecassis, K., Rodriguiz, R.M., Huang, X.-P., Norval, S., Sassano, M.F., Shin, A.I., Webster, L.A., Simeons, F.R.C., Stojanovski, L., Prat, A., Seidah, N.G., Constam, D.B., Bickerton, G.R., Read, K.D., Wetsel, W.C., Gilbert, I.H., Roth, B.L., Hopkins, A.L.: Automated design of ligands to polypharmacological profiles. Nature **492**(7428), 215–220 (2012). doi:10.1038/nature11691
8. Miyao, T., Kaneko, H., Funatsu, K.: Inverse qspr/qsar analysis for chemical structure generation (from y to x). Journal of Chemical Information and Modeling **56**(2), 286–299 (2016). doi:10.1021/acs.jcim.5b00628
9. Churchwell, C.J., Rintoul, M.D., Martin, S., Jr., D.P.V., Kotu, A., Larson, R.S., Sillerud, L.O., Brown, D.C., Faulon, J.-L.: The signature molecular descriptor: 3. inverse-quantitative structure–activity relationship of icam-1 inhibitory peptides. Journal of Molecular Graphics and Modelling **22**(4), 263–273 (2004). doi:10.1016/j.jmgm.2003.10.002
10. Wong, W.W., Burkowski, F.J.: A constructive approach for discovering new drug leads: Using a kernel methodology for the inverse-qsar problem. J Cheminform **1**, 4–4 (2009). doi:10.1186/1758-2946-1-4
11. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Interspeech, vol. 2, p. 3 (2010)
12. Eck, D., Schmidhuber, J.: A first look at music composition using lstm recurrent neural networks. Technical report (2002)
13. Segler, M.H.S., Kogej, T., Tyrchan, C., Waller, M.P.: Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks. ArXiv e-prints (2017). 1701.01329
14. Gómez-Bombarelli, R., Duvenaud, D.K., Hernández-Lobato, J.M., Aguilera-Iparraguirre, J., Hirzel, T.D., Adams, R.P., Aspuru-Guzik, A.: Automatic chemical design using a data-driven continuous representation of molecules. CoRR **abs/1610.02415** (2016)
15. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. CoRR **abs/1609.05473** (2016)
16. Sutton, R., Barton, A.: Reinforcement Learning: An Introduction, 1st edn. MIT Press, Cambridge, MA (1998)
17. Jaques, N., Gu, S., Turner, R.E., Eck, D.: Tuning recurrent neural networks with reinforcement learning. CoRR **abs/1611.02796** (2016)
18. Leo, A., Hansch, C., Elkins, D.: Partition coefficients and their uses. Chemical Reviews **71**(6), 525–616 (1971). doi:10.1021/cr60274a001
19. Bickerton, G.R., Paolini, G.V., Besnard, J., Muresan, S., Hopkins, A.L.: Quantifying the chemical beauty of drugs. Nature Chemistry **4**, 90–98 (2012). doi:10.1038/nchem.1243
20. Gaulton, A., Bellis, L.J., Bento, A.P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., Overington, J.P.: Chembl: a large-scale bioactivity database for drug discovery. Nucleic Acids Res **40**, 1100–1107 (2012). doi:10.1093/nar/gkr777. Version 22
21. Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y.: An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. ArXiv e-prints (2013). 1312.6211
22. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, MA (2016)
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural

Comput. **9**(8), 1735–1780 (1997). doi:10.1162/neco.1997.9.8.1735
24. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv e-prints (2014). 1412.3555
25. SMILES. Accessed 7 April 2017. http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html
26. RDKit: open source cheminformatics. Version: 2016-09-3. http://www.rdkit.org/
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
28. Tensorflow. Version: 1.0.1. http://www.tensorflow.org
29. Sun, J., Jeliazkova, N., Chupakin, V., Golib-Dzib, J.-F., Engkvist, O., Carlsson, L., Wegner, J., Ceulemans, H., Georgiev, I., Jeliazkov, V., Kochev, N., Ashby, T.J., Chen, H.: Excape-db: an integrated large scale dataset facilitating big data analysis in chemogenomics. Journal of Cheminformatics **9**(1), 17 (2017). doi:10.1186/s13321-017-0203-5
30. Jaccard, P.: Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin del la Société Vaudoise des Sciences Naturelles **37**, 547–579 (1901)
31. Rogers, D., Hahn, M.: Extended-connectivity fingerprints. Journal of Chemical Information and Modeling **50**(5), 742–754 (2010). doi:10.1021/ci100050t
32. Butina, D.: Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets. Journal of Chemical Information and Computer Sciences **39**(4), 747–750 (1999). doi:10.1021/ci9803381
33. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011). Version 0.17
34. Reutlinger, M., Koch, C.P., Reker, D., Todoroff, N., Schneider, P., Rodrigues, T., Schneider, G.: Chemically Advanced Template Search (CATS) for Scaffold-Hopping and Prospective Target Prediction for 'Orphan' Molecules. Mol Inform **32**(2), 133–138 (2013)
35. Senger, S.: Using Tversky similarity searches for core hopping: finding the needles in the haystack. J Chem Inf Model **49**(6), 1514–1524 (2009)
36. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. CoRR **abs/1409.2329** (2014)
37. Wan, L., Zeiler, M., Zhang, S., LeCun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28. ICML'13, pp. 1058–1066 (2013)
38. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)

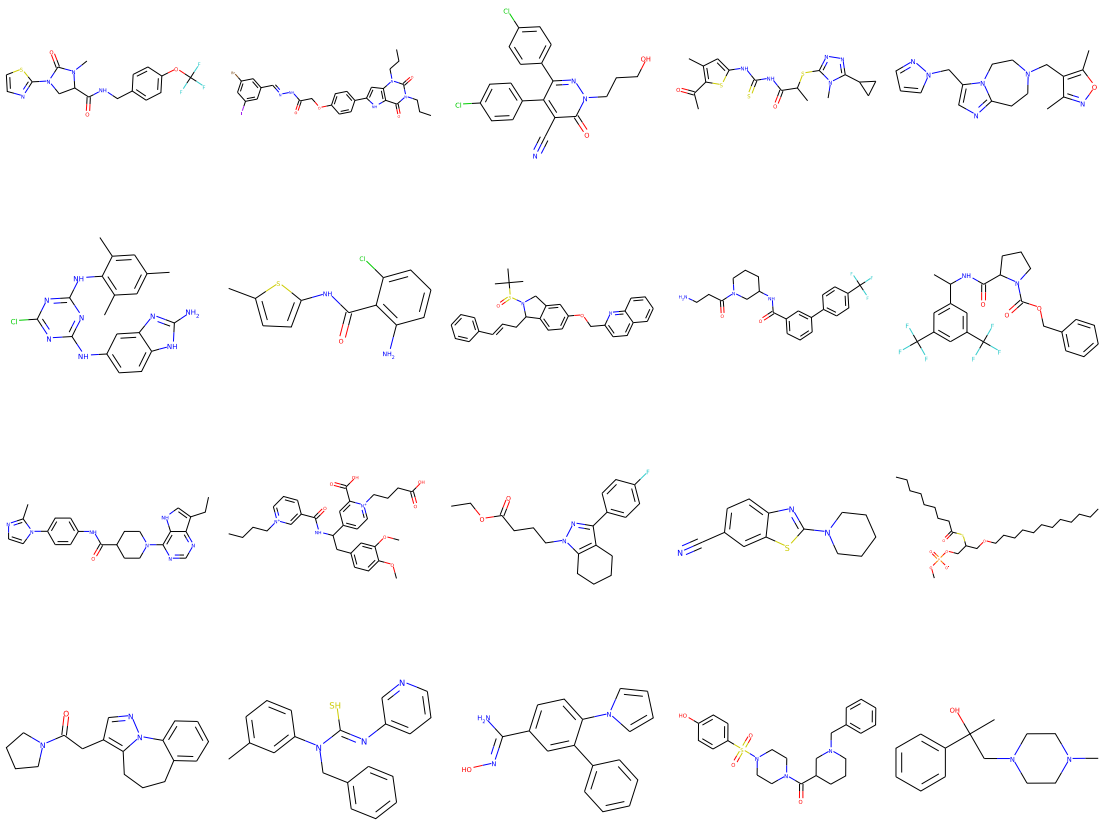

Figure 10 Appendix 1. Randomly selected structures generated by the canonical Prior.

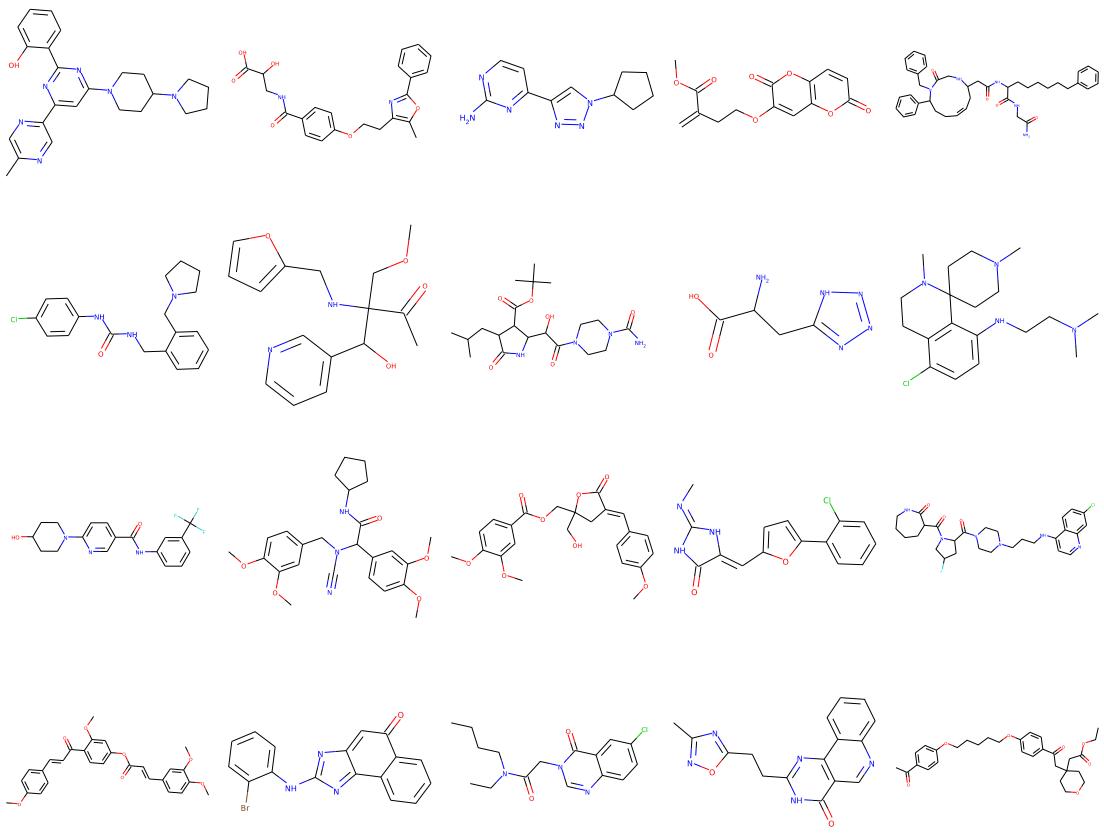

Figure 11 Appendix 2. Randomly selected structures generated by the Agent trained to avoid sulphur.

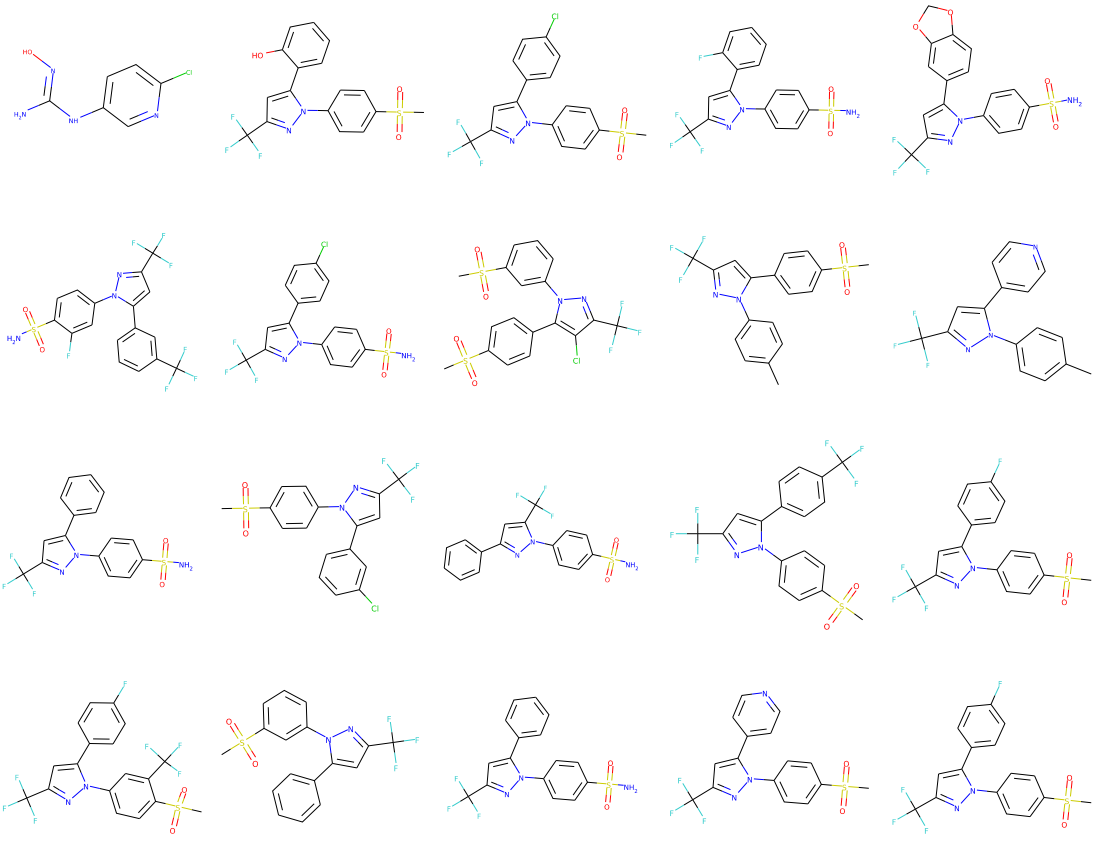

Figure 12 Appendix 3. Randomly selected structures generated by the Agent based on the reduced Prior trained to design analogues of Celecoxib.

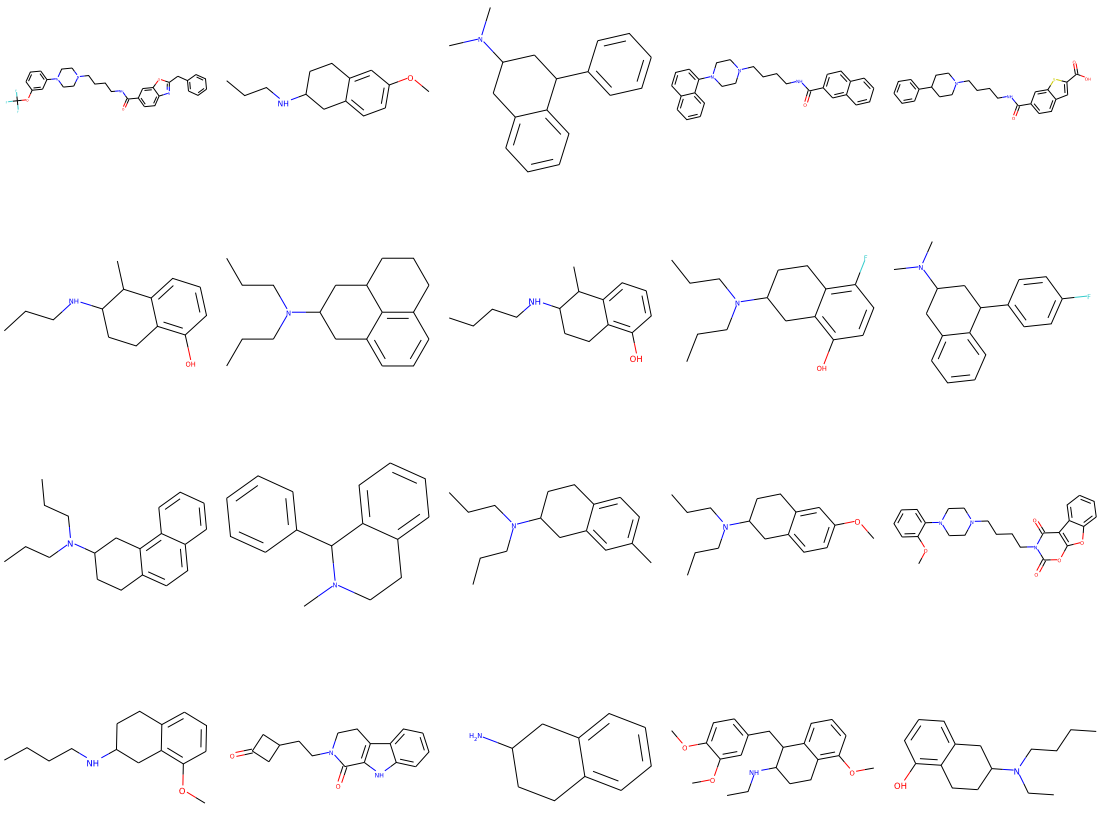

Figure 13 Appendix 4. Randomly selected structures generated by the Agent based on the reduced Prior trained to design actives against DRD2.