REINVENT 2.0 – an AI tool for de novo drug design

Thomas Blaschke[€], Josep Arús-Pous[§ ⊥], Hongming Chen[¥], Christian Margreitter[§], Christian Tyrchan[‖], Ola Engkvist[§], Kostas Papadopoulos[§], Atanas Patronov[§].*

§ Hit Discovery, Discovery Sciences, MolecularAI, R&D, AstraZeneca Gothenburg, Sweden

‖ Medicinal Chemistry, BioPharmaceuticals Early RIA, R&D, AstraZeneca, Gothenburg, Sweden

⊥ Department of Chemistry and Biochemistry, University of Bern, Freiestrasse 3, 3012 Bern, Switzerland.

¥ Chemistry and Chemical Biology Centre, Guangzhou Regenerative Medicine and Health-Guangdong Laboratory, Science Park, Guangzhou, China

€ Department of Life Science Informatics, B-IT, LIMES Program Unit Chemical Biology and Medicinal Chemistry, Rheinische Friedrich-Wilhelms-Universität, Endenicher Allee 19c, D-53115 Bonn, Germany * Corresponding author: atanas.patronov@astrazeneca.com

Abstract

In the past few years, we have witnessed a renaissance of the field of de novo drug design. The advancements in deep learning and artificial intelligence (AI) have triggered an avalanche of ideas about how to translate such techniques to a variety of domains including the field of drug design. A range of architectures have been devised to find the optimal way of generating chemical compounds by using either graph or SMILES based representations. With this application note we aim to offer the community a production-ready tool for de novo design, named REINVENT. It can be effectively applied on drug discovery projects that are striving to resolve either exploration or exploitation problems while navigating the chemical space. It can facilitate the idea generation process by bringing to the researcher's attention the most promising compounds. REINVENT's code is publicly available at https://github.com/MolecularAI/Reinvent

Introduction

The main goal of de novo drug design is to identify novel active compounds that can simultaneously satisfy a constellation of essential optimization goals such as activity, selectivity, physical-chemical and ADMET properties. Because of the sheer number of possible solutions, it is a non-trivial task to optimally satisfy such a multitude of requirements which makes the search process slow and costly even when it is only conducted in silico. Therefore, having an efficient solution which enables the navigation of chemical space and generation of relevant ideas is essential. To address such needs the research community has recently turned its focus towards artificial intelligence (AI) based generative models that are capable of proposing promising small molecules. The potential of generative models for chemical space exploration has been demonstrated in numerous studies [1]–[3]. Various neural network architectures have been engineered and a plethora of AI training strategies have been employed in the race to device more efficient methods for the generation of compounds. A number of architectures, such as Variational Autoencoders (VAEs) [4], [5], Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) cells [6], Conditional RNNs or Generative Adversarial Networks have been proven successful in generating molecules by using data representation of molecules either as molecular graphs or SMILES [7]–[10].

However, while all of these architectures and many others are provided as open-source, only a few [11] are in a state that allows to readily apply the code on drug design related problems without the need of spending a significant amount of time developing missing functionalities. Ideally, users should be able to navigate the chemical space efficiently in two general use cases: exploration and exploitation mode. For exploitation, users define an area of interest and focus on generating compounds that share similar structural features. In contrast, the exploration mode enables them to obtain compounds that share less structural similarity but still satisfy other desired features. This implies the necessity to utilize not only predictive models and structure similarity/dissimilarity but also various rule-based scoring components to push towards or pull away from specific areas of the chemical space. Moreover, to be able to adapt appropriately to any given drug discovery project at hand, the ability to fine-tune each of these potential scoring function components is paramount.

To achieve such behavior, apart from having a deep learning architecture with a reliable generative potential, it is essential to provide an efficient navigation mechanism. In order to address such needs, we are describing in the current paper the latest version of REINVENT - our in-house developed tool for de novo design of small molecules.

Application Overview

In its core, REINVENT is using a generative model with an architecture derived from the work of Arus-Pous et al [12]. The model is trained on a dataset derived from ChEMBL [13] and capable of generating compounds in the SMILES format. It has been trained by “randomizing” the SMILES representation of the input data, which is essentially a data augmentation technique [12]. Randomizing the compounds’ representation uses multiple SMILES encodings for the same compound, thus ensuring that the model will likely learn the grammar rather than memorizing specific strings or parts of them. The resulting model shows a significantly improved generalization potential and produces SMILES strings with validity of above 99% [12]. Uniform sampling of the chemical space by the model is a prerequisite for efficient exploration. The model can generate random valid compounds and is able to dive into any region of that space for exhaustive exploitation [12]. However, we are mostly interested in compounds that only act on a specific target and such creative potential of generative models is of little practical use unless specific context is given. Therefore, it is often necessary to direct the generative model towards relevant areas in the chemical space that contain compounds of interest. We achieve this by subjecting it to a Reinforcement Learning (RL) [14] scenario while aiming to satisfy a set of requirements that could vaguely sketch the desired compounds. In other words, the generative model will try to maximize the outcome of a scoring function that contains multiple components/parameters, thus computing an MPO score [15]. To stir the generation of compounds towards the desired direction, REINVENT employs a composite scoring function consisting of different user-defined components. Each component is responsible for a simple target property. The feedback from the scoring function is used in a RL loop with a policy iteration as described by Olivecrona et al. [16]. Two RNNs are used in an actor-critic scenario where the critic is a prior RNN (Prior) that remains constant and serves as a baseline thus guaranteeing that the knowledge of SMILES syntax will be retained. The actor (which we will refer to as the Agent) can be an identical copy of the Prior or a focused version that has already undergone some training. The Agent takes actions by sampling a batch of SMILES **S** which are evaluated by the Prior and scored by the scoring function. The resulting score is combined with Prior's likelihood and used to form the augmented likelihood (eq 1). The augmented likelihood essentially sets the bar for the Agent since the loss is calculated as the squared difference between the Agent's likelihood and the augmented likelihood (eq 2). Also, **σ** is a scalar value, automatically adjusted to guarantee a proper margin between augmented and Agent's likelihood values.

$$logP(\boldsymbol{S})_{Augmented} = logP(\boldsymbol{S})_{Prior} + \boldsymbol{\sigma} * MPO(\boldsymbol{S})_{score} \tag{1}$$

$$loss = \left[logP(\boldsymbol{S})_{Augmented} - logP(\boldsymbol{S})_{Agent}\right]^2 \tag{2}$$

The RL scenario is complemented with an inception feature which can speed up the focusing of the Agent. Inception is essentially an extended experience replay [17] that allows users to pre-incept SMILES of interest so that the RL run generates compounds within an area of interest in a fewer number of steps. This can be particularly useful for specific exploitation scenarios.

Another key feature that has influence over the RL driven training of the Agent is the diversity filter as it can penalize the frequent generation of similar compounds. Each of these features are discussed in further detail below.

Scoring Functions

REINVENT offers two general scoring function formulations (equations 3 and 4). The individual components of the scoring function can be either combined as a weighted sum or as a weighted product [18]. The individual score components can have different weight coefficients reflecting their importance in the overall score. Score contribution from each component can vary in the range between 0 and 1. As a result, the overall score is also within a range of 0 to 1.

$$P(X) = \left[\prod_i p(x_i)^{w_i}\right]^{1/\sum_i w_i} \tag{3}$$

$$S(X) = \frac{\sum_i w_i * p(x_i)}{\sum_i w_i} \tag{4}$$

The scoring function can be comprised of components such as physical-chemical properties, predictive models (both regression and classification), shape similarity, Tanimoto similarity, and Jaccard distance scores [19]. The predictive model component in REINVENT works with both regression and classification types of scikit-learn [20] predictive models. XGB Regressor model types from the xgboost python library can be also employed [21]. A notable limitation of the current implementation is that only fingerprint descriptors can be used. REINVENT supports various representations of ECFP descriptors, MACCS keys and Avalon descriptors all of which are implemented in the RDKit library [22]–[25]. Classification models are expected to be binary class predictors and the corresponding probability of belonging to the positive class is used as an output. However, the regression models can output any continuous value and a suitable transformation should be applied to scale the predictions into the required [0, 1] interval. We offer a variety of transformation functions, including sigmoid and double sigmoid as well as step functions for transforming non-continuous components such as the number of hydrogen bond donors and acceptors. We also provide a custom interpolation transformation where the score is transformed by a function that interpolates between user-defined pairs of minima and maxima. The choice of transformation depends on the predicted property or the calculated descriptor. For properties that are only desirable to lie within a certain range we would seek to apply a double sigmoid transformation to cap the score between the preferred lower and upper bound values resulting in a score of 0 outside and increasing up to 1.

Combining multiple components in a single scoring function presents a typical multiparameter optimization problem. By increasing the number of components, the probability of discovering solutions that achieve maximum score can drop significantly as the different components are likely to pull the MPO score in different directions. Another key aspect is the cumulative uncertainty resulting from the use of multiple predictive models at a time. Even if nearly perfect models are used, combining them will result in geometric amplification of the uncertainty in the outcome of a weighted product formulation. While such an effect would be milder in the weighted sum scenario, the exploration of the chemical space by using multiple predictive models could still be likened to navigating at sea with a slightly broken compass.

In addition to the standard version (eq 3), we offer custom weighted scoring function formulations (eqs 5 and 6) where P_{MS} is a Matching Substructure (MS) component and P_{CA} is a Custom Alerts (CA) component. These two are binary penalty components. MS can be used to focus the generation of compounds towards a specific scaffold of interest. It uses a list of SMARTS [26] as an input and it penalizes the overall score if none of the desired substructures is represented in the generated compound. MS produces a score of either 1 or 0.5 depending on whether the scaffold is present or not, thereby being quite helpful for exploitation scenarios. CA can be either 0 or 1 and it also uses a list of SMARTS patterns that normally capture undesired moieties in the generated compounds. If there is a match with any of the listed alerts the overall score will be 0 thus penalizing the future generation of similar compounds. CA can be used also for scaffold hopping if the user is aiming for novelty and wants to avoid certain molecular substructures.

$$P(\mathrm{X}) = P(X)_{MS} \times P(X)_{CA} \times \left[\prod_i p(x_i)^{w_i}\right]^{\frac{1}{\sum_i w_i}} \tag{5}$$

$$S(X) = P(X)_{MS} \times P(X)_{CA} \times \left[\frac{\sum_i w_i * p(x_i)}{\sum_i w_i}\right] \tag{6}$$

Another common use case is to try to optimize against a target of interest while simultaneously minimizing the probability of binding to one or more off-targets. For this scenario, we offer a Selectivity Component (SC). SC works with two predictive models: one is used to predict the target activity and another for predicting an off-target activity. If both predictive models are regression type, the difference between the predicted activities Δ (eq 7) is calculated and consequently subjected to a sigmoid transformation thus forming the SC score. In cases where one of the models is a classifier the regression model prediction is first subjected to transformation and the resulting Δ is output as an SC score.

$$\Delta = P_{activity} - P_{off-target} \tag{7}$$

In cases where $\Delta < 0$, we assign a lower cap of 0.01 since producing 0 for the component would result in a 0 overall score if used with equations 3 or 5 and will not be sufficiently informative for the Agent. Multiple SC can be used when multiple off-targets are possible.

If properly formulated the scoring function will most likely guide the Agent towards a narrow niche of the chemical space, such that yields high MPO scores. As a result, the Agent will become extremely focused over time and ultimately sample only a handful of compounds with high probability. At this stage the scoring function will reach a plateau, the diversity of the generated structures will be minimal and conducting any further RL steps will not yield any new results. In order to generate another batch of novel compounds, we would need to start over and climb the same learning curve over multiple RL steps in order to optimize the scoring function. However, there is no guarantee that the RL process will not converge in a similar chemical space as the previous run. To enforce generative diversity and stimulate the exploration of a broader chemical space, we employ an additional feature in the RL loop - the Diversity Filters (**DF**).

Diversity Filters

DF can be regarded as a collection of buckets that are used for keeping track of all generated scaffolds and the compounds that share those scaffolds. Obviously, not all generated compounds are of interest and only those that are scored by the MPO function above a certain threshold will enter the scaffold buckets. Once a compound with a score above the threshold has been generated, its scaffold is extracted and stored in a scaffold registry and the compound enters the corresponding bucket. The buckets have limited capacity and once the limit of compounds in a given bucket has reached the allowed threshold, any subsequent bucket affiliation will be penalized. Every new compound that enters a full bucket will be assigned a score of zero thus informing the Agent that this area of chemical space has become unfavorable. It is important to note that compounds will be added to the bucket even if the bucket limit has been exceeded. The only impact will be on the Agent, since it will be constantly discouraged from producing similar compounds that share a given scaffold. This will enforce the Agent to seek alternative solutions thus achieving in effect chemical space exploration and will prevent the Agent from becoming stuck in local minima and thus generating the same compounds repeatedly. All collected compounds are kept and stored until the end of the RL run and become available as a csv formatted file.

Users can select their diversity strategy by using Topological DF, Identical Murcko DF or a Scaffold Similarity DF [27]. The Topological DF is the most restrictive since it is agnostic of the atom types. It is created by removing all side chains and subsequently converting all atoms in the structure to sp3 carbons. The other two DF also remove all side chains but retain the atom types. Identical Murcko DF only checks if there is a bucket with exactly the same scaffold while Scaffold Similarity is more permissive and can include compounds into the bucket if they satisfy a certain threshold of scaffold similarity.

Directing the generative process.

Once the Agent starts generating compounds of sufficiently high MPO score we can define that it has reached a **state of productivity**. However, starting from a random point in the chemical space and slowly focusing the Agent to a state where it can generate compounds of interest can be a complex and time-consuming task. It could even prove to be an impossible task within a single RL run, especially if the MPO formulation is too complex and has multiple components. To overcome this and to speed up the overall RL process we have identified two approaches that can complement each other.

Transfer Learning (TL)

At the beginning of the RL process, the Agent is an identical copy of the Prior. It possesses the same generative capacity and the potential to sample compounds from rather vast area of the chemical space. While this holds a great promise, it can also be an efficiency overhead since for the Agent to become “productive” will first need to find a “rewarding” chemical space. This will have an impact on the RL search for both types of problems: exploitation and exploration, particularly when the MPO score includes multiple conflicting components. To overcome it and to speed up the overall RL process, we resort to pre-focusing the Prior by conducting a TL with a small dataset of compounds sharing features of relevance to our problem. Once focused, the Agent will have an increased probability of sampling a chemical subspace of interest. Such generated compounds will be rewarded higher by the scoring function providing more specific directionality to the generative process. We can use the focused Agent as a starting point for the RL instead of using a copy of the Prior and reach sooner to a **state of productivity**.

Inception

Another way to speed up the transition and reach the **state of productivity** is by "incepting" compounds that are ranked highly by the scoring function and represent the chemical space of interest. Inception is used in analogy to experience replay in the RL loop. Compounds that we know would be scored highly are introduced to the inception memory before beginning the RL process. At each RL step, a fraction of the inception memory is randomly sampled and is added to the set of compounds generated by the Agent. In this way, early in the RL process, the Agent is presented with highly scoring compounds and will be driven to focus towards the chemical subspace defined by the inception compounds. Also, it will reach a **state of productivity** sooner. While helpful in the early stages of the training, replaying the compounds that scored well can lead to a very focused Agent. This is particularly likely if the RL run is sufficiently long. For exploration goals it would be best to use it in conjunction with DF since it will prevent excessive focusing by down scoring the repetitive ideas. Inception memory has a limited size and the compounds that are scored lower will be forgotten. The incepted compounds should be ideally from the chemical space of interest with a score below the DF threshold so that they are not discarded instantly.

Inception could be considered as a substitute to TL. However, it is far less efficient in terms of RL steps as it takes longer on average to reach productivity when comparing to starting from a focused Prior state.

Both, Inception and pre-focusing with TL complement each other. We recommend using inception for problems where reaching the **state of productivity** seems impossible or might take too long due to a very complex scoring function. Another frequent use case is when the number of compounds available for focusing is insufficient to train efficiently a focused prior with TL. In these situations, having a handful of compounds that score well and are frequently presented to the Agent can significantly help to reach the relevant chemical space within a reasonable time.

Logging

Essential for monitoring of the learning process is the availability of a comprehensive logging system. In REINVENT we utilize Tensorboard [28] to provide information about the evolution of the Agent during TL by sampling after each step and displaying the likelihood distribution for the sampled data. Stats on validity of the smiles and the most frequently encountered molecules are also shown. For RL we are plotting the evolution of the scoring function and the individual scoring component contributions to the overall score. We are also displaying the highest scoring compounds after each RL step. As an alternative, we also provide the implementation used by us for remote logging which can be set up to post the logging results to a custom REST endpoint.

Implementation

REINVENT is an open-source Python application. It uses PyTorch 1.3.0 [29] as a deep learning engine and RDKit version 2019.03.3.0 [30] as a chemistry engine. It works exclusively with scikit-learn based machine learning models and for the detailed logging of the chemical space navigation process, it makes use of Tensorboard's implementation in PyTorch.

Conclusion

We have described a production-ready, open-source application for de novo generation of small molecules. It can be used to address both exploration and exploitation type of problems while allowing a flexible formulation of complex MPO scores. Examples of various use cases are provided with the code repository.

Apart from providing a ready-to-use solution, with releasing the code, we are hoping to facilitate the research on using generative methods for drug discovery. We also hope that it can be used as an interaction point for future scientific collaborations.

References

[1] J. Arús-Pous *et al.*, “SMILES-Based Deep Generative Scaffold Decorator for De-Novo Drug Design,” Jan. 2020, doi: 10.26434/CHEMRXIV.11638383.V1.

[2] A. Zhavoronkov *et al.*, “Deep learning enables rapid identification of potent DDR1 kinase inhibitors,” *Nat. Biotechnol.*, vol. 37, no. 9, pp. 1038–1040, Sep. 2019, doi: 10.1038/s41587-019-0224-x.

[3] M. H. S. Segler, T. Kogej, C. Tyrchan, and M. P. Waller, “Generating focused molecule libraries for drug discovery with recurrent neural networks,” *ACS Cent. Sci.*, vol. 4, no. 1, pp. 120–131, Jan. 2018, doi: 10.1021/acscentsci.7b00512.

[4] T. Blaschke, M. Olivecrona, O. Engkvist, J. Bajorath, and H. Chen, “Application of Generative Autoencoder in *De Novo* Molecular Design,” *Mol. Inform.*, vol. 37, no. 1–2, p. 1700123, Jan. 2018, doi: 10.1002/minf.201700123.

[5] R. Gómez-Bombarelli *et al.*, “Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules,” *ACS Cent. Sci.*, vol. 4, no. 2, pp. 268–276, Feb. 2018, doi: 10.1021/acscentsci.7b00572.

[6] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[7] D. Weininger, “SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules,” *J. Chem. Inf. Comput. Sci.*, vol. 28, no. 1, pp. 31–36, Feb. 1988, doi: 10.1021/ci00057a005.

[8] O. Prykhodko *et al.*, “A de novo molecular generation method using latent vector based generative adversarial network,” *J. Cheminform.*, vol. 11, no. 1, p. 74, Dec. 2019, doi: 10.1186/s13321-019-0397-9.

[9] P.-C. Kotsias, J. Arús-Pous, H. Chen, O. Engkvist, C. Tyrchan, and E. J. Bjerrum, “Direct Steering of de novo Molecular Generation using Descriptor Conditional Recurrent Neural Networks (cRNNs),” Nov. 2019, doi: 10.26434/CHEMRXIV.9860906.V2.

[10] Y. Li, L. Zhang, and Z. Liu, “Multi-objective de novo drug design with conditional graph generative model,” *J. Cheminform.*, vol. 10, no. 1, p. 33, Dec. 2018, doi: 10.1186/s13321-018-0287-6.

[11] R. Winter, F. Montanari, A. Steffen, H. Briem, F. Noé, and D. A. Clevert, “Efficient multi-objective molecular optimization in a continuous latent space,” *Chem. Sci.*, vol. 10, no. 34, pp. 8016–8024, Aug. 2019, doi: 10.1039/c9sc01928f.

[12] J. Arús-Pous *et al.*, “Randomized SMILES strings improve the quality of molecular generative models,” *J. Cheminform.*, vol. 11, no. 1, Nov. 2019, doi: 10.1186/s13321-019-0393-0.

[13] A. Gaulton *et al.*, “The ChEMBL database in 2017,” *Nucleic Acids Res.*, vol. 45, no. D1, pp. D945–D954, Jan. 2017, doi: 10.1093/nar/gkw1074.

[14] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction Second edition, in progress.”

[15] T. T. Wager, X. Hou, P. R. Verhoest, and A. Villalobos, “Moving beyond rules: The development of a central nervous system multiparameter optimization (CNS MPO) approach to enable alignment of druglike properties,” *ACS Chem. Neurosci.*, vol. 1, no. 6, pp. 435–449, Jun. 2010, doi: 10.1021/cn100008c.

[16] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, “Molecular de-novo design through deep reinforcement learning,” *J. Cheminform.*, vol. 9, no. 1, p. 48, Dec. 2017, doi: 10.1186/s13321-017-0235-x.

[17] L.-J. Lin, “Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching,” 1992.

[18] D. J. Cummins and M. A. Bell, “Integrating Everything: The Molecule Selection Toolkit, a System for Compound Prioritization in Drug Discovery,” *J. Med. Chem.*, vol. 59, no. 15, pp. 6999–7010, Aug. 2016, doi: 10.1021/acs.jmedchem.5b01338.

[19] T. T. . Tanimoto, *An elementary mathematical theory of classification and prediction by T.T. Tanimoto | National Library of Australia*. .

[20] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python,” *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[21] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, vol. 13-17-August-2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[22] D. Rogers and M. Hahn, “Extended-connectivity fingerprints,” *J. Chem. Inf. Model.*, vol. 50, no. 5, pp. 742–754, May 2010, doi: 10.1021/ci100050t.

[23] P. Gedeck, B. Rohde, and C. Bartels, “QSAR - How good is it in practice? Comparison of descriptor sets on an unbiased cross section of corporate data sets,” *J. Chem. Inf. Model.*, vol. 46, no. 5, pp. 1924–1936, Sep. 2006, doi: 10.1021/ci050413p.

[24] H. L. Morgan, “The Generation of a Unique Machine Description for Chemical Structures—A Technique Developed at Chemical Abstracts Service,” *J. Chem. Doc.*, vol. 5, no. 2, pp. 107–113, May 1965, doi: 10.1021/c160017a018.

[25] J. L. Durant, B. A. Leland, D. R. Henry, and J. G. Nourse, “Reoptimization of MDL Keys for Use in Drug Discovery,” *ACS Publ.*, vol. 42, no. 6, pp. 1273–1280, Nov. 2002, doi: 10.1021/ci010132r.

[26] “Daylight Theory: SMARTS - A Language for Describing Molecular Patterns.” [Online]. Available: https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html. [Accessed: 12-Feb-2020].

[27] G. W. Bemis and M. A. Murcko, “The properties of known drugs. 1. Molecular frameworks,” *J. Med. Chem.*, vol. 39, no. 15, pp. 2887–2893, Jul. 1996, doi: 10.1021/jm9602928.

[28] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.”

[29] A. Paszke *et al.*, “Automatic differentiation in PyTorch.”

[30] L. G., “RDKit.” [Online]. Available: http://www.rdkit.org/. [Accessed: 12-Feb-2020].