Supporting Information

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, R&D, AstraZeneca Gothenburg, Pepparedsleden 1, 43183, Sweden

‖ Medicinal Chemistry, Early RIA, Biopharmaceuticals R&D, AstraZeneca, Gothenburg, Pepparedsleden 1, 43183, 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, 510530, Guangzhou, China

* Corresponding author: atanas.patronov@astrazeneca.com

General use cases

With regards to the specific project needs we distinguish two general groups of idea generating behavior - exploration and exploitation. While in exploration mode we are trying to identify a broader set of ideas, in exploitation we aim to find solutions that are close to each other. The diversity of the proposed solutions may be judged by different, problem specific criteria. However, it is important to note that we should not tightly link exploitation to goal-directed generation and exploration to distribution-learning as both can be achieved by using either strategies. For example, RL can be used to achieve both types of behavior. Since the purpose of this application note is simply to introduce a de novo design tool rather than diving into the specifics of a concrete problem we will only suggest the general steps that need to be taken to achieve the relevant behavior. We are however sharing a number of concrete examples that could be found in [Reinvent Community].

Distribution learning

Distribution learning is represented by workflow "B" in ***table S2*** where a small dataset is introduced to the model and then from the model a large dataset is sampled. As a result, this requires screening through vast amounts of data which may consist of hundreds of thousands of molecules or even millions. Normally drug designing involves optimization of more than one parameter and in addition to conducting multiple evaluations of low scoring compounds the user risks to still fail to identify a suitable set of compounds. This may require another step of transfer learning, sampling and evaluation. While the workflow of conducting transfer learning followed by some form of scoring is a completely valid approach and seems to be widely advocated we do however find this to be a less efficient and much more computationally demanding method in comparison to the goal directed generation.

Goal-directed learning

In REINVENT, the goal directed generation is achieved with workflows “C”, “D”, “E” and “F” from ***table S2***, all of which are carried out by using reinforcement learning (RL). RL leads the model in multiple steps towards an area of the chemical space that yields sufficient reward. Compounds are generated at every step of the process and evaluated. The batches are relatively small (default size 128). The users can define a preferred score threshold in Diversity Filters [1] (DF) (default 0.4). All generated compounds above this score are collected in the DF memory. If the collected compounds are repeatedly generated the score is penalized thus pushing away the generative model and stimulating it to generate different compounds. If a DF is set to work at any other mode than “NoFilter”, score penalty will be applied according to the particular filter chosen. This will further enforce the generative model towards diversity and will induce exploratory behavior as the model is encouraged to find structurally diverse compounds that still satisfy the score. ***Figure S4*** aims to illustrate these behaviors.

It is important to highlight that the compound generation is done during the process of RL and not after. The resulting agent is no longer needed. The data is collected not at the end of the reinforcement learning but during. This is a notable difference between TL and RL as we do not use the end state of the model. We only use the data harvested during the RL journey.

Features

Inception

Inception is essentially a modified version of experience replay which is a well-known concept applied on variety of problems including playing games [2–4]. The purpose of experience replay is to "memorize" the compounds from each RL step. Since it has a limited memory size (defined by the user), only the best compounds are retained in the memory once the limit is reached. The memory is updated at each step with better compounds pushing away the worst. The purpose of the experience replay is to randomly select a small batch of previously generated compounds that have been memorized and to present it to the agent. This serves as "reminder" of which SMILES were more successful and speeds up the learning process. The modification we introduce here, enables loading compounds that we know would be scored highly before beginning with the RL process. We can estimate how they would score by using the "Scoring" mode of REINVENT with the same scoring function as intended for the RL. At each RL step, a fraction of the inception memory is randomly sampled and 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 incepted compounds. This will speed up the learning in the early stages and will help the agent to reach the **state of productivity** sooner. Since this is only an application note we do not provide any supporting data on how inception is influencing the RL. However, we provide a number of examples in [Reinvent Community] where inception is used. We intend to demonstrate its properties in a separate study.

Scoring Functions

To provide the user with sufficient amount of freedom we supply two general scoring function formulations (**eqs S1 and S2**). The individual components of the scoring function can be either combined as a weighted sum or as a weighted product [5]. The individual score components can have different weight coefficients reflecting their importance in the overall score. Score contribution from each component can vary in a [0,1] range. As a result, the overall score is also within the same [0,1] range. In the equations below the score for sequence ***x*** is denoted ***S*** and is either a weighted product (eq S2) or a weighted sum (eq S1). The user-selected components are denoted as ***p*** in both equations.

$$S(x) = \left[\prod_i p_i(x)^{w_i}\right]^{1/\sum_i w_i} \tag{S1}$$

$$S(x) = \frac{\sum_i w_i * p_i(x)}{\sum_i w_i} \tag{S2}$$

The scoring function can be comprised of components such as physico-chemical properties, predictive models (both regression and classification), shape similarity, Tanimoto similarity, and Jaccard distance scores [6]. The full list of components included in this release can be found in ***table S3***. The predictive model component in REINVENT works with both regression and classification types of scikit-learn [7] predictive models. XGB Regressor model types from the xgboost python library can be also employed [8]. In the current release the predictive models used by REINVENT work with various fingerprint descriptors that are implemented within RDKit library [9–12] such as ECFP descriptors, MACCS keys and Avalon. For continuous descriptors reflecting the physico-chemical properties we do not offer implementations as of yet. However, the descriptor pool can be easily extended. Classification models are expected to be binary class predictors and the corresponding probability of belonging to the positive class is used as input. 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 be constrained 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. While we cannot speculate on the advantages of a different formulation, the fact that the components vary in a [0,1] range suggests that a poorly scoring component with a high weight coefficient would result in a lower score when calculated as a weighted product (eq **S1**) as an alternative to the weighted sum (eq **S2**). Another key aspect is the cumulative uncertainty resulting from the use of multiple predictive models at a time. Putting a higher weight to such components would also amplify this uncertainty. Therefore, the exploration of the chemical space by using multiple predictive models would resemble navigating at sea with a slightly broken compass. Since, there may be other components that also introduce uncertainty in the scoring function (such as docking, which is a feature not yet released) the users should be mindful of how the weights are assigned and how trustworthy is the calculated score. There is a significant amount of freedom to define the scoring function by selecting each individual component and assigning weights to them. There is no uniform recipe to follow. Normally higher weights are assigned to components for which it is desirable to have more impact on the score. Since, the components may contradict each other the maximum possible value will be in some point of equilibrium for the included components. If this equilibrium point is too low (for instance below the productivity threshold defined in DF, above which we start collecting compounds) then it might appear that the model failed to learn whereas in reality the scoring function was poorly set.

It is important to bear in mind that the generative models can learn to exploit weaknesses of the score objectives, which is a well-known fact, documented in various experiments summed up by Lehman, J. (2019) [13]. This is commonly solved by re-defining the objectives in a way that prevents such weaknesses from being rewarded.

In addition to the standard version (**eqs S1 and S2**), we offer custom weighted scoring function formulations (**eqs S3 and S4**) 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 which is calculated by using the implementation of Bemis-Murcko scaffolds in RDKit [14]. It uses a list of SMARTS [15] as input and 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. We provide examples with pre-set CA that penalize certain moieties. However, the proposed content might be irrelevant to the users so they should revise and readjust the SMARTS patterns so that they suit their needs.

$$S(x) = P(x)_{MS} \times P(x)_{CA} \times \left[\prod_i p_i(x)^{w_i}\right]^{\frac{1}{\sum_i w_i}} \tag{S3}$$

$$S(x) = P(x)_{MS} \times P(x)_{CA} \times \left[\frac{\sum_i w_i * p_i(x)}{\sum_i w_i}\right] \tag{S4}$$

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 S5**) 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.

$$\Delta(x) = P(x)_{activity} - P(x)_{off-target} \tag{S5}$$

In cases where Δ < 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 S1 or S3 and will not be sufficiently informative for the agent. This is because, in the case of a complex scoring function the generative model aims to find the equilibrium between all components. Receiving 0 as a result from one of the components in the scoring function for the weighted product scenario will set the overall score to 0 for each compound that failed for a single criterion. Assuming that this component rewards only a rare event this will result in a variety of different SMILES strings all labeled as 0 for various reasons. Also, this will obliterate all the information coming from the other components. In contrast providing a non-zero score will give the agent sufficiently informative response and will help to learn from it. In addition, if the other components are picked to work in synergy this will help the agent to still maximize the response from them and find solutions that also satisfy the failing component. It is quite possible that returning 0 may even prevent the agent from learning anything within the scope of the reinforcement learning run. Multiple SC can be used when multiple off-targets are possible.

Figures

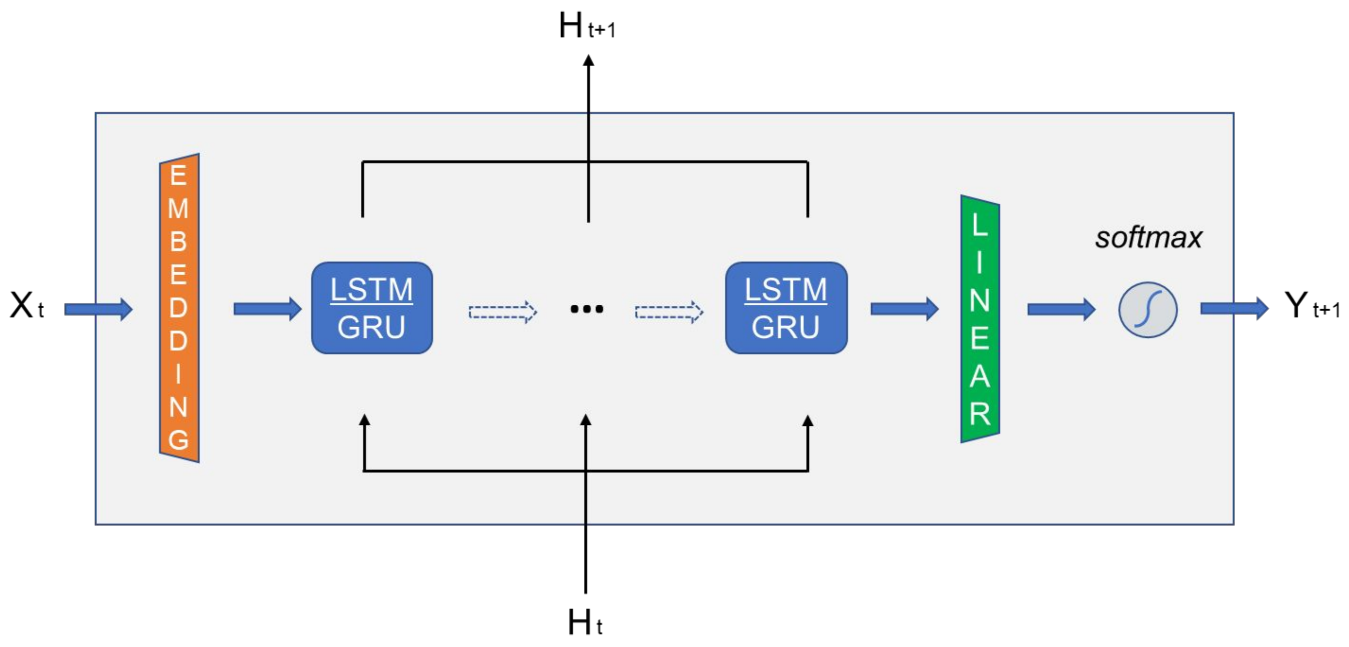

Figure S1. General architecture of the generative model within REINVENT. For step ***t*** the input one-hot encoded vector $\boldsymbol{X_t}$ is passed through an embedding layer. The size of the embedding layer should be lower than the following LSTM/GRU cells. The users can build generative models with a variable number of either LSTM or GRU layers. The specific settings with which the example model provided in the repository is created can be found here [Reinvent Community]. The size of the linear layer is defined by the vocabulary size which in turn is determined by identifying all unique tokens when processing the supplied dataset. The softmax transformation provides token probability distribution. $\boldsymbol{H_t}$ is the input hidden state at step ***t***.

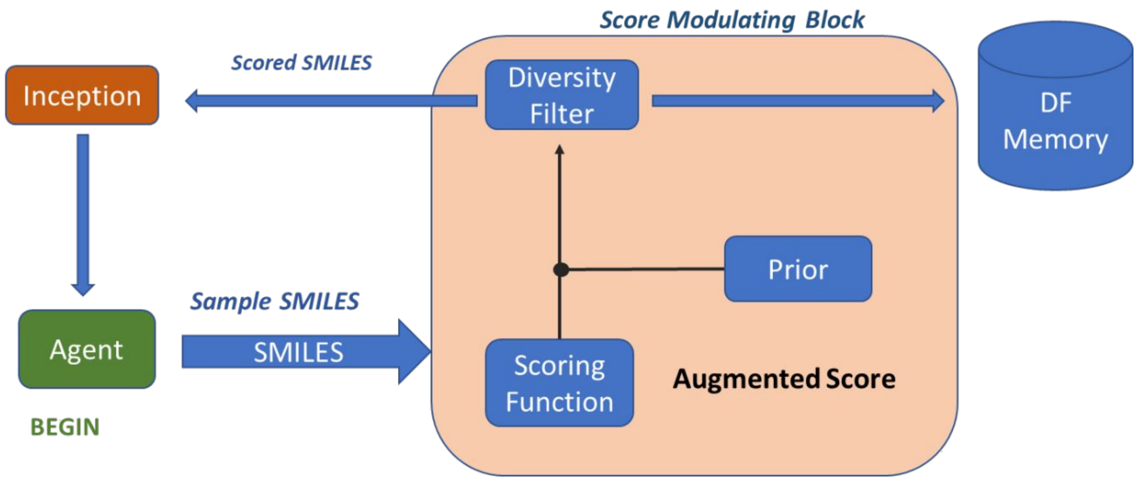

Figure S2. The reinforcement learning cycle. The agent can be either a focused or general prior. Sampled smiles are evaluated by the scoring function and the score is combined with the priors negative log-likelihood to form an Augmented score, which is subsequently used to calculate the agent's loss. Diversity filter **(DF)** is collecting all unique smiles that have a score above a user-defined threshold. At the end of the RL run, all collected SMILES in the DF memory are outputted to a file. DF penalizes compounds with a scaffold that has been generated too frequently. Inception stores the best suggestions from each cycle and adds previously suggested SMILES strings that have scored highly.

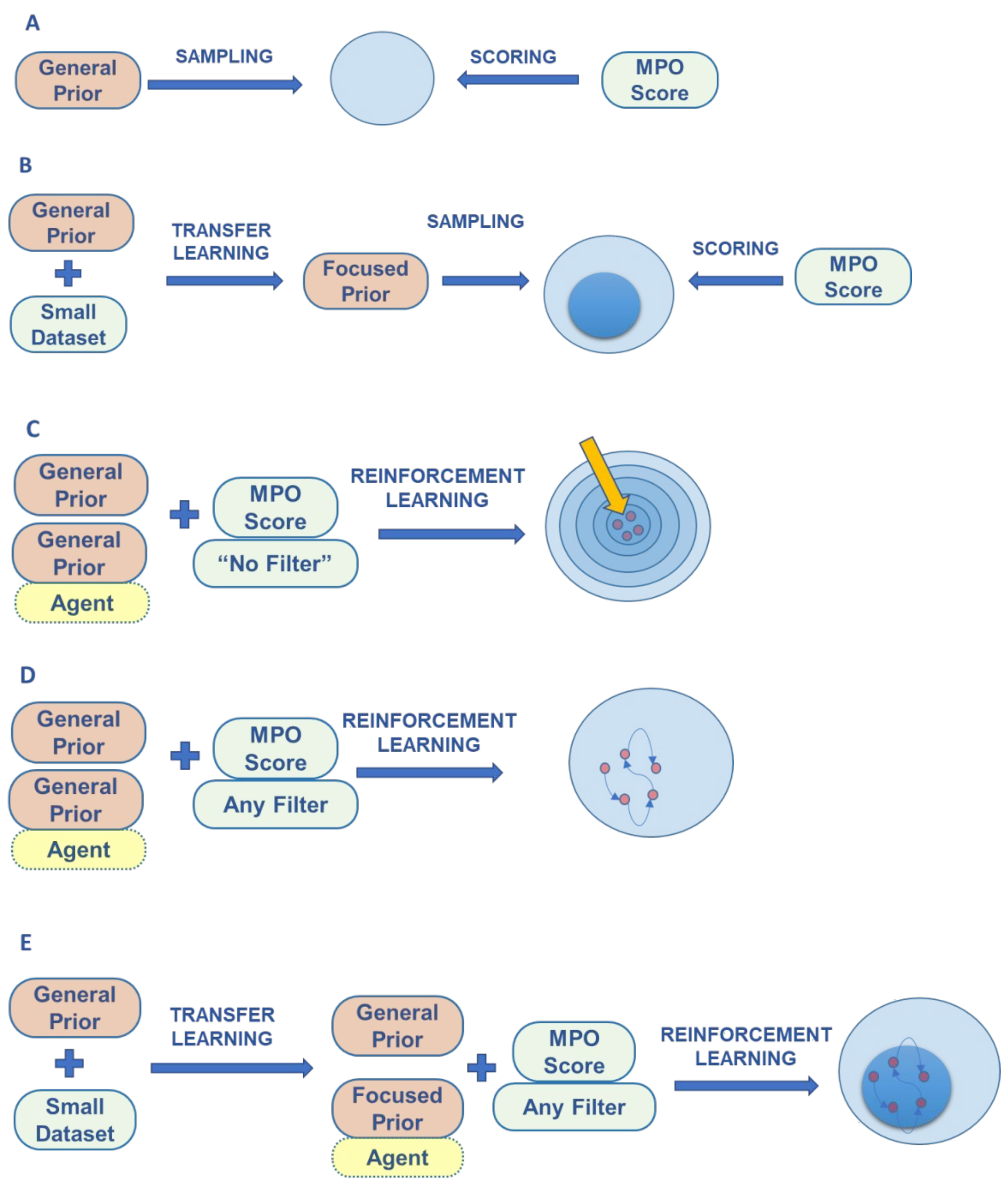

Figure S3. General use cases supported by REINVENT. **A)** Sampling followed by scoring. **B)** Transfer learning with a small dataset containing compounds of interest. The resulting "focused prior" can generate certain areas of the chemical space with a higher probability than others. Transfer learning is followed by sampling and then scoring. **C)** Reinforcement learning with a scoring function as an objective and DF set to "NoFilter" option. This scenario trains the agent to maximize the score from the scoring function and during the process of training collects smiles. The data acquired in the course of training belongs to the regions with a high MPO score illustrated by red circles. The resulting agent is discarded. There is no diversity enforced and it is expected that generated solutions would be linked to a confined chemical space. That is, if we estimate the diversity by the number of chemical scaffolds found. **D)** Same as C) but with DF to enforce diversity. This will lead to a larger number of chemical scaffolds proposed. The generation of ideas is also done during the RL run. **E)** We start with initial transfer learning with a small dataset thus generating a “focused prior”. We use the resulting “focused prior” as an agent in the subsequent reinforcement learning. This way the initial state of the agent will be biased towards the training set which we used during the “focusing” step with transfer learning. Generation of ideas is also done during the reinforcement learning run.

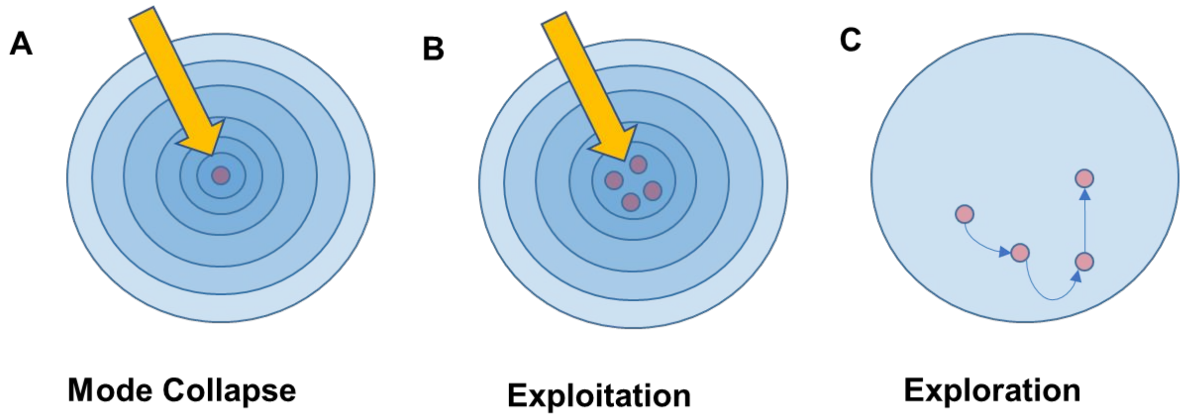

Figure S4. *A* – Depicts an end case where the agent is optimizing a scoring function thus gradually becoming more focused which means that it generates more frequently similar compounds. This behavior is possible when DFs are not present in the RL loop. Without DFs RL can result into a mode collapse which produces the same compounds over and over. This is particularly true for longer runs with very narrow scoring objectives (similarity to a specific compound for e.g.). *B* – Illustrates a reinforcement learning scenario where the agent is optimizing a scoring function in a small region of the chemical space. This behavior is achieved with relatively short runs where “NoFilter” chosen as a DF option or by increasing the bucket size of the other DF types. *C* – Illustrates exploration behavior induced by including the exploration promoting DF. DF keep the history of all generated compounds that are ranked above a user-defined threshold thus modulating the score and resulting in a constantly changing scoring function. The changing goal forces the agent to generate novel ideas thus resulting in exploration of the chemical space. Red circles represent the regions of compounds with high score while the blue area is part of the chemical space containing compounds with a lower MPO score.

Tables

Table S1: List of all currently available running modes. Details on how to execute the specific runs are provided in [Reinvent Community].

RUNNING MODE	DESCRIPTION
CREATE MODEL	**This running mode allows the user to create a naïve generative model that has not been trained yet. The purpose is to parse the training set once and extract all available tokens so that the vocabulary is created. The empty model is used as an input for TRANSFER LEARNING mode along with the training set to train a new generative model.**
TRANSFER LEARNING	**TL can be used to either train an entirely new generative model (essentially by applying "teachers forcing") or to introduce an existing model to a smaller set of compounds. This will increase the likelihood of generating similar structures.**
REINFORCEMENT LEARNING	**This is a mode that enables goal-directed generation. It is important to stress the fact that the novel ideas are generated throughout the entire run. The resulting model at the end of the run is normally discarded. RL can be used either in exploration or in exploitation mode.**
SAMPLING	**Allows to sample smiles from the generative model.**
SCORING	**Scoring is useful for evaluating SMILES strings by applying a user-defined scoring function.**

Table S2: General use cases supported by REINVENT. For cases "C", "D" and "F" at the beginning of the run the prior and the agent are essentially the same generative model whereas in case "E" the agent is a "focused" version of the prior.

LABEL	WORKFLOW	USE CASE	DESCRIPTION
A	Sampling, followed by scoring	Exploration	Sampling directly from analogous generative models has shown that novel compounds can be generated in large numbers [16,17]. Subsequently evaluating these compounds with a scoring function resembles scenario where a static database of compounds is screened. Since it may easily involve millions of compounds being sampled without yielding any good scores this does not seem to be a particularly efficient approach. This becomes even less appealing if the scoring function contains very expensive components such as docking, pharmacophore similarity, or slow predictive models with computationally expensive descriptors.
B	Transfer Learning with a small set of compounds, followed by sampling and then scoring	Exploration	The generative model is subjected to transfer learning with a smaller set of compounds that are relevant to the project of interest. This will bias the resulting model to produce project specific compounds with much higher probability than any random compounds [18,19]. Therefore much smaller dataset can suffice to find good hits when using the scoring function. Still, there is no guarantee of a good score and this might require additional sampling iterations and possibly additional transfer learning steps.
C	Reinforcement Learning with Diversity Filters set to "NoFilter" or other filters with extended bucket size	Exploitation	This is a mode that enables goal-directed generation. It is important to stress the fact that the novel ideas are generated throughout the entire run. The resulting model at the end of the run is normally discarded. "NoFilter" mode allows to collect solutions that might be very similar due to the fact that no scaffold dissimilarity is enforced. Extending the bucket size when using other DFs also promotes exploitation since it allows larger number of similar compounds to be sampled before starting to penalize.
D	Reinforcement Learning with Diversity Filters set to different than "NoFilter" type	Exploration	This is also a mode that enables goal-directed generation. However, using Diversity Filters with any type of scaffold filter will enforce generation of molecules that have different scaffolds. While the scoring function definition remains the same the diversity filters will memorize the discovered so far compounds and will penalize same or similar suggestions. As a result, the agent will adapt by producing novel scaffolds while also aiming to maximize the scoring function.

			Diversity Filters can be also seen as an adaptive penalty that modulates the score. This method of exploration is by far the most efficient since it brings the evaluation of random compounds to a minimum.
E	Transfer Learning with Subsequent Reinforcement Learning	Exploitation and Exploration	This is a strategy that can be used in order to speed up the learning process by reaching sooner a state of productivity of the generative model. This is achieved by conducting transfer learning with a set of compounds that have the desired properties. For example, if we aim to maximize a predictive model among the other components we would use all the compounds that are considered as active by this model. If we aim towards certain subseries of compounds we would only use those that share the specific features for transfer learning. After concluding with transfer learning the resulting agent is “focused” on the specific set. The users can conduct TL for multiple epochs and observe the stats from each epoch thus deciding which agent is focused enough. The tensorboard log provides various information including the most frequently sampled compounds so that the user can visually inspect if the desired structural patterns are present. The state of the trained agent is stored after each epoch and the user can choose which agent to use for the subsequent RL. Note that this is done to pre-condition the agent for the RL so that it has a head-start in the exploitation search. Although this is more likely to be used for exploitation, the user can still use it as a starting point for exploration by choosing the appropriate diversity filter types. The model should be still the “unfocused” generative model.
F	Reinforcement Learning with Inception	Exploitation and Exploration	This is analogous to case “E”. Instead of “focusing” the agent the users can load in the inception memory a list of compounds that share desirable properties. The agent will become directly exposed to those examples during the RL and the scoring function will reward them thus stimulating the agent to further generate similar compounds. Users should incept compounds that are scored high enough by the scoring function and at the same time are below the “minscore” threshold of the chosen DF.

Table S3: List of all currently available scoring function components.

COMPONENT NAME	DESCRIPTION
PREDICTIVE PROPERTY	**Uses scikit-learn library for predictive models. Works with both classification and regression models. Essentially the models should follow the library's interface. Please, consult with the provided examples in [Reinvent Community]. Any model object that has the methods "predict()" and "predict_proba()" should be compatible.**
TANIMOTO SIMILARITY	**Requires a user defined set of smiles and returns the highest similarity score to the provided set.**
JACCARD DISTANCE	**Requires a user defined set of smiles and returns the lowest distance score to the provided set.**
MATCHING SUBSTRUCTRE	**Requires a user defined set of SMARTS. This is a penalty component. Returns 1 if there is a substructure match and 0.5 otherwise.**
CUSTOM ALERTS	**Requires a user defined set of SMARTS patterns indicating unwanted moieties. This is a penalty component. Returns 0 if there is a match and 1 otherwise.**
QED SCORE	**Uses the QED implementation in RDKit.**
MOLECULAR WEIGHT	**Phys-Chem property calculated by RDKit.**
TPSA	**Phys-Chem property calculated by RDKit.**
ROTATABLE BONDS	**Phys-Chem property calculated by RDKit.**
NUMBER OF HYDROGEN BOND DONOROS	**Phys-Chem property calculated by RDKit.**
NUMBER OF RINGS	**Phys-Chem property calculated by RDKit.**
SELECTIVITY	**Uses two scikit-learn models. Works with both classification and regression models. One model is predicting the target activity and the other is providing**

	an off-target prediction. The score is reflecting a user defined activity gap between the target and the off-target predictions.

Table S4: List of all currently available Diversity Filters. All modes apart from "No Filter" enhance the exploration to a variable extend

RUNNING MODE	DESCRIPTION
NO FILTER	**No filter mode is used. Compounds are simply stored and no penalty is applied on the scores. This mode is used for Exploitation.**
IDENTICAL MURCKO DIVERSITY FILTER	**All side chains are stripped and the remaining scaffold is used to identify whether there is already an existing bucket or a new bucket should be created.**
SCAFFOLD SIMILARITY DIVERSITY FILTER	**This filter uses the same method for generating the scaffold as Identical Murcko. It however includes compounds to the given bucket if they are with scaffold similarity above a user defined threshold. Usage of the Identical Murcko filter is actually a special case, where the similarity threshold is precisely 1. The score is calculated by comparing the atom pair fingerprint between the scaffolds [20] and using Tanimoto similarity.**
IDENTICAL TOPOLOGICAL DIVERSITY FILTER	**This is the most restrictive filter since it is agnostic of the atom type. This is achieved by removing all side chains and converting all atoms in the remaining scaffold to sp3 carbons.**

References

(1) Blaschke, T.; Engkvist, O.; Bajorath, J.; Chen, H. *Memory-Assisted Reinforcement Learning for Diverse Molecular de Novo Design*; ChemRxiv, 2020. https://doi.org/10.26434/CHEMRXIV.12693152.V1.

(2) Lin, L.-J. *Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching*; 1992; Vol. 8.

(3) Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*; International Conference on Learning Representations, ICLR, 2016.

(4) Wang, Z.; Bapst, V.; Heess, N.; Mnih, V.; Munos, R.; Kavukcuoglu, K.; de Freitas, N. Sample Efficient Actor-Critic with Experience Replay. *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.* **2016**.

(5) Cummins, D. J.; Bell, M. A. Integrating Everything: The Molecule Selection Toolkit, a System for Compound Prioritization in Drug Discovery. *J. Med. Chem.* **2016**, *59* (15), 6999–7010. https://doi.org/10.1021/acs.jmedchem.5b01338.

(6) Tanimoto, T. T. . *An Elementary Mathematical Theory of Classification and Prediction by T.T. Tanimoto | National Library of Australia*.

(7) 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, É. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

(8) Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; Association for Computing Machinery: New York, New York, USA, 2016; Vol. 13-17-August-2016, pp 785–794. https://doi.org/10.1145/2939672.2939785.

(9) Rogers, D.; Hahn, M. Extended-Connectivity Fingerprints. *J. Chem. Inf. Model.* **2010**, *50* (5), 742–754. https://doi.org/10.1021/ci100050t.

(10) Gedeck, P.; Rohde, B.; Bartels, C. QSAR - How Good Is It in Practice? Comparison of Descriptor Sets on an Unbiased Cross Section of Corporate Data Sets. *J. Chem. Inf. Model.* **2006**, *46* (5), 1924–1936. https://doi.org/10.1021/ci050413p.

(11) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures—A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5* (2), 107–113. https://doi.org/10.1021/c160017a018.

(12) Durant, J. L.; Leland, B. A.; Henry, D. R.; Nourse, J. G. Reoptimization of MDL Keys for Use in Drug Discovery. *ACS Publ.* **2002**, *42* (6), 1273–1280. https://doi.org/10.1021/ci010132r.

(13) Lehman, J.; Clune, J.; Misevic, D.; Adami, C.; Altenberg, L.; Beaulieu, J.; Bentley, P. J.; Bernard, S.; Beslon, G.; Bryson, D. M.; Chrabaszcz, P.; Cheney, N.; Cully, A.; Doncieux, S.; Dyer, F. C.; Ellefsen, K. O.; Feldt, R.; Fischer, S.; Forrest, S.; Frénoy, A.; Gagné, C.; Goff, L. Le; Grabowski, L. M.; Hodjat,

B.; Hutter, F.; Keller, L.; Knibbe, C.; Krcah, P.; Lenski, R. E.; Lipson, H.; MacCurdy, R.; Maestre, C.; Miikkulainen, R.; Mitri, S.; Moriarty, D. E.; Mouret, J.-B.; Nguyen, A.; Ofria, C.; Parizeau, M.; Parsons, D.; Pennock, R. T.; Punch, W. F.; Ray, T. S.; Schoenauer, M.; Shulte, E.; Sims, K.; Stanley, K. O.; Taddei, F.; Tarapore, D.; Thibault, S.; Weimer, W.; Watson, R.; Yosinski, J. The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. *Artif. Life* **2018**, 1–33.

(14) Bemis, G. W.; Murcko, M. A. The Properties of Known Drugs. 1. Molecular Frameworks. *J. Med. Chem.* **1996**, *39* (15), 2887–2893. https://doi.org/10.1021/jm9602928.

(15) Daylight Theory: SMARTS - A Language for Describing Molecular Patterns https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (accessed Feb 12, 2020).

(16) Arús-Pous, J.; Johansson, S. V.; Prykhodko, O.; Bjerrum, E. J.; Tyrchan, C.; Reymond, J. L.; Chen, H.; Engkvist, O. Randomized SMILES Strings Improve the Quality of Molecular Generative Models. *J. Cheminform.* **2019**, *11* (1), 71. https://doi.org/10.1186/s13321-019-0393-0.

(17) Arús-Pous, J.; Blaschke, T.; Ulander, S.; Reymond, J. L.; Chen, H.; Engkvist, O. Exploring the GDB-13 Chemical Space Using Deep Generative Models. *J. Cheminform.* **2019**, *11* (1), 20. https://doi.org/10.1186/s13321-019-0341-z.

(18) Moret, M.; Friedrich, L.; Grisoni, F.; Merk, D.; Schneider, G. Generative Molecular Design in Low Data Regimes. *Nat. Mach. Intell.* **2020**, *2* (3), 171–180. https://doi.org/10.1038/s42256-020-0160-y.

(19) Merk, D.; Friedrich, L.; Grisoni, F.; Schneider, G. De Novo Design of Bioactive Small Molecules by Artificial Intelligence. *Mol. Inform.* **2018**, *37* (1). https://doi.org/10.1002/minf.201700153.

(20) Smith, D. H.; Carhart, R. E.; Venkataraghavan, R. Atom Pairs as Molecular Features in Structure-Activity Studies: Definition and Applications. *J. Chem. Inf. Comput. Sci.* **1985**, *25* (2), 64–73. https://doi.org/10.1021/ci00046a002.