Deep Reinforcement Learning for *De-Novo* Drug Design

Mariya Popova[1,2,3], Olexander Isayev[1*], Alexander Tropsha[1*]

[1]Laboratory for Molecular Modeling, Division of Chemical Biology and Medicinal Chemistry, UNC Eshelman School of Pharmacy, University of North Carolina, Chapel Hill, NC, 27599, USA.

[2]Moscow Institute of Physics and Technology, Dolgoprudny, Moscow region, 141700, Russia.

[3]Skolkovo Institute of Science and Technology, Moscow, 143026, Russia.

*Correspondence: A.T (alex_tropsha@unc.edu), O.I. (olexandr@olexandrisayev.com).

Abstract

We propose a novel computational strategy based on deep and reinforcement learning techniques for *de-novo* design of molecules with desired properties. This strategy integrates two deep neural networks – generative and predictive – that are trained separately but employed jointly to generate novel chemical structures with the desired properties. Generative models are trained to produce chemically feasible SMILES, and predictive models are derived to forecast the desired compound properties. In the first phase of the method, generative and predictive models are separately trained with supervised learning algorithms. In the second phase, both models are trained jointly with reinforcement learning approach to bias newly generated chemical structures towards those with desired physical and biological properties. In this proof-of-concept study, we have employed this integrative strategy to design chemical libraries biased toward compounds with either maximal, minimal, or specific ranges of physical properties, such as melting point and hydrophobicity, as well as to develop novel putative inhibitors of JAK2. This new approach can find general use for generating targeted chemical libraries optimized for a single desired property or multiple properties.

Introduction

Artificial intelligence (AI) systems can radically transform the practice of scientific discovery(*1*). The combination of big data and artificial intelligence has been referred to as the fourth industrial revolution(*2*). Artificial intelligence is also revolutionizing medicine(*3*) including radiology, pathology and other medical specialties (*4*, *5*). Deep Learning (DL) technologies are beginning to find applications in drug discovery (*6*–*8*) including areas of molecular docking (*9*), transcriptomics (*10*) , reaction mechanism elucidation (*11*) , and molecular energy prediction (*12*, *13*).

The drug discovery pipeline is notoriously sequential (*14*). Hits from a high throughput screen (HTS) are slowly progressed toward promising lead compounds. Next, ADMET and selectivity profiles are optimized with a challenge to maintain potency and efficacy. High failure rates in late-stage compound development and clinical trials could potentially be avoided if models that forecast compound fate could be employed at the initial molecular design phase. One example of such an approach is the broad use of Lipinski's rules of bioavailability (*15*, *16*) to filter molecules that possess the desired bioactivity *in vitro*. Indeed, it has been acknowledged that the broad use of these rules has substantially reduced the failure rate in experimental ADME studies of drug candidates (*17*). The crucial step in many new drug discovery projects is the formulation of a well-motivated hypothesis for new lead compound generation (*de novo* design) or compound selection from available or synthetically feasible chemical libraries based on the available SAR data. Commonly, an interdisciplinary team of scientists generates the new hypothesis by employing computational models of drug action and relying on their expertise and medicinal chemistry intuition. Therefore, the design hypothesis is often biased towards preferred chemistry (*18*) or driven by model interpretation (*19*).

Automated approaches for designing compounds with desired properties de novo have become an active field of research in the last 15 years (*20*, *21*). In an attempt to design new compounds, both medicinal and computational chemists face virtually infinite chemical space. The diversity of synthetically feasible chemicals that can be considered as potential drug-like molecules is estimated to be between 10^{30} and 10^{60} (*22*, *23*). Great advances in both computational algorithms(*24*, *25*), hardware, and high-throughput screening (HTS) technologies (*16*) notwithstanding, the size of this virtual library prohibits its exhaustive sampling and testing by systematic construction and evaluation of each individual compound. Local optimization approaches have been proposed but they do not ensure the optimal solution, as the design process converges on a local or 'practical' optimum by stochastic sampling, or restrict the search to a defined section of chemical space which can be screened exhaustively (*20*, *26*–*28*). Notably, a method for exploring chemical space based on continuous encodings of molecules was proposed recently (*29*). It allows efficient, directed gradient-based search through chemical space but does not include biasing libraries toward special physical or biological properties. Another very recent approach for generating focused molecule libraries with the desired bioactivity using Recurrent Neural Networks was proposed as well (*30*). However, properties of produced molecules could not be controlled well. Another approach proposed for novel molecular development is adversarial autoencoder (*31*), which suggests a new method for virtual screening of large libraries, but cannot design novel molecules. Points from the latent space are projected to the nearest known molecule in the screening database.

Herein, we propose a novel method for generating chemical compounds with desired physical, chemical and/or bioactivity properties that is based on deep reinforcement learning (RL). Reinforcement learning is a subset of artificial intelligence which is used to solve dynamic decision problems. It involves the analysis of possible actions, estimation of the statistical relationship between the actions and their possible outcomes, followed by a determination of a treatment regime that attempts to find the most desirable outcome based on the analysis The integration of reinforcement learning and neural networks dates back to 1990s (*32*). However, with the recent advancement of deep learning (DL), benefiting from Big Data, new powerful algorithmic approaches are emerging. There is a current renaissance of RL (*33*), especially when it is combined with deep neural networks, i.e., deep reinforcement learning. Most recently, RL was employed to achieve superhuman performance in the game Go (*34*), considered practically intractable due to the theoretical complexity of over 10^{140} possible solutions (*35*). One may see an analogy with the complexity of chemical space exploration with an algorithm that avoids brute-force computing to examine every possible solution. Below we describe application of deep RL to the problem of designing chemical libraries with the desired properties.

Results

We have devised a novel RL based *de novo* design method for generating chemical compounds with desired physical, chemical or bioactivity properties. We termed it as ReLeaSE (<u>Re</u>inforcement <u>Lea</u>rning for Structural Evolution). The general ReLeaSE workflow (Figure 1) is represented by two deep neural networks (generative *G* and predictive *P*). The process of training consists of two stages. During the first stage, both models are trained separately with supervised learning algorithms, and during the second stage, the models are trained jointly with a reinforcement learning approach that optimizes target properties. In this system, the generative model plays the role of an agent whereas the predictive model plays the role of the critic, which estimates the agent's behavior by assigning a numerical reward to every generated molecule. The reward is a function of the numerical property generated by the predictive model. The generative model is trained to maximize the expected reward.

Reinforcement learning formulation. Both generative *G* and predictive model *P* are combined into one RL system. The set of actions *A* is defined as an alphabet, i.e., the entire collection of letters and symbols used to define canonical SMILES strings (*36*) that are most commonly used to encode chemical structures. A molecule of aspirin, for example, is encoded as [CC(=O)OC1=CC=CC=C1C(=O)O]. The set of states *S* is defined as all possible strings in the alphabet with lengths from 0 to some value *T*. The state s_0 with length 0 is unique and considered to be the initial state. The state s_T of length *T* is called the terminal state, as it causes training to end. The subset of terminal states $S^* = \{s_T \in S\}$ of *S* which contains all states s_T with length *T* is called the terminal states set. Reward $r(s_T)$ is calculated at the end of the training cycle when the terminal state is reached. Intermediate rewards $r(s_t),\ t < T$ are equal to 0. In these terms, the generator network *G* can be treated as a policy approximation model. At each time step *t*, $0 < t < T$, *G* takes the previous state s_{t-1} as an input and estimates probability distribution $p(a_t \mid s_{t-1})$ of the next action. Afterwards, the next action a_t is sampled from this estimated probability. Reward $r(s_T)$ is a function of the predicted property of s_T by the predictive model *P*:

$$r(s_T) = f\big(P(s_T)\big) \tag{1}$$

where *f* is chosen depending on the task. Some examples of the functions *f* are provided in the computational experiment section. Given these notations and assumptions, the problem of generating chemical compounds with desired properties can be formulated as a task of finding a vector of parameters *Θ* of policy network *G* which maximizes the expected reward:

$$J(\Theta) = \mathbb{E}[r(s_T) \mid s_0, \Theta] = \sum_{s_T \in S^*} p_\Theta(s_T) r(s_T) \rightarrow max. \tag{2}$$

This sum iterates over the set S^* of terminal states. In our case this set is exponential and the sum cannot be computed exactly. According to the law of large numbers, we can approximate this sum as a mathematical expectation by sampling terminal sequences from the model distribution:

$$J(\Theta) = \mathbb{E}[r(s_T) \mid s_0, \Theta] = \mathbb{E}_{a_1 \sim p_\Theta(a_1 \mid s_0)} \mathbb{E}_{a_2 \sim p_\Theta(a_2 \mid s_1)} \cdots \mathbb{E}_{a_T \sim p_\Theta(a_T \mid s_{T-1})} r(s_T). \tag{3}$$

To estimate *J(Θ)*, we sequentially sample a_t from the model *G* for t from 0 to *T*. The unbiased estimation for *J(Θ)* is the sum of all rewards in every time step, which, in our case, equals the reward for the terminal state as we assume that intermediate rewards are equal to 0. This quantity needed to be maximized; therefore, we need to compute its gradient. This can be done, for example, with a REINFORCE algorithm (*37*) that uses the approximation of mathematical expectation as a sum, which we provided in equation 3, and the following form:

$$\partial_\Theta f(\Theta) = f(\Theta)\frac{\partial_\Theta f(\Theta)}{\partial \Theta} = f(\Theta)\partial_\Theta \log f(\Theta). \tag{4}$$

Therefore, the gradient of *J(Θ)* can be written down as:

$$\begin{aligned}\partial_\Theta J(\Theta) &= \sum_{s_T \in S^*} \left[\partial_\Theta p_\Theta(s_T)\right] r(s_T) = \sum_{s_T \in S^*} p_\Theta(s_T) \left[\partial_\Theta \log p_\Theta(s_T)\right] r(s_T) \\ &= \sum_{s_T \in S^*} p_\Theta(s_T) \left[\sum_{t=1}^{T} \partial_\Theta \log p_\Theta(a_t | s_{t-1})\right] r(s_T) \\ &= \mathbb{E}_{a_1 \sim p_\Theta(a_1|s_0)} \mathbb{E}_{a_2 \sim p_\Theta(a_2|s_1)} \dots \mathbb{E}_{a_T \sim p_\Theta(a_T|s_{T-1})} \left[\sum_{t=1}^{T} \partial_\Theta \log p_\Theta(a_t | s_{t-1})\right] r(s_T),\end{aligned} \tag{5}$$

which gives an algorithm $\partial_\Theta J(\Theta)$ estimation.

Neural networks architectures. Model *G* (Figure 1A) is a generative recurrent neural network(*38*, *39*), which outputs molecules in SMILES notation. We use a special type stack-augmented recurrent neural network (Stack-RNN) (*40*).

Regular recurrent neural networks like LSTM (*41*) and GRU (*42*) are unable to solve the sequence prediction problems due to their inability to count. One of the challenging examples of sequences that cannot be properly modeled by regular recurrent networks are words from the Dyck language, a language of balanced strings of brackets (*43*, *44*). Another weakness of regular recurrent neural networks is their inability to capture long term dependencies, which leads to difficulties in generalizing to longer sequences (*45*). All of these properties are required to learn the language of the SMILES notation. In a valid SMILES molecule, in addition to correct valence for all atoms, one must count, ring opening and closure, as well as bracket sequences with several bracket types. Therefore, Stack RNNs are the proper choice for modeling such sequence dependencies.

The Stack-RNN defines a new neuron or cell structure on top of the standard GRU cell (See Figure 1A). It has two additional multiplicative gates referred to as the memory stack, which allow the Stack-RNN to learn meaningful long-range interdependencies. Stack is a differentiable structure onto and from which continuous vectors are inserted and removed. In stack terminology, the insertion operation is called `PUSH` operation and the removal operation is called `POP` operation. These traditionally discrete operations are continuous here, since `PUSH` and POP operations are permitted to be real values in the interval (0, 1). Intuitively, we can interpret these values as the degree of certainty with which some controller wishes to `PUSH` a vector *v* onto the stack, or `POP` the top of the stack.

The second model P is a predictive model (see Figure 1D) for estimating physical, chemical or biological properties of molecules. This property prediction model is a deep neural network, which consists of an embedding layer (*46*), LSTM layer and two dense layers. This network is designed to calculate user-specified property (activity) of the molecule taking SMILES string as an input data vector. In a practical sense, this learning step is analogous to traditional Quantitative Structure-Activity Relationships (QSAR) models. However, unlike conventional QSAR, no numerical descriptors are needed, as the model distinctly learns directly from the SMILES notation.

Generation of chemicals with novel structures. The generator network was trained with ~1.5M structures from ChEMBL21 database (*47*) (Please see Methods for technical details). To demonstrate the versatility of the baseline (unbiased) Stack RNN, we generated a dataset of over one million (1M) virtually synthesized compounds. All structures are available for download from Supplementary Information. Random examples of the generated compounds are illustrated in Figure 2.

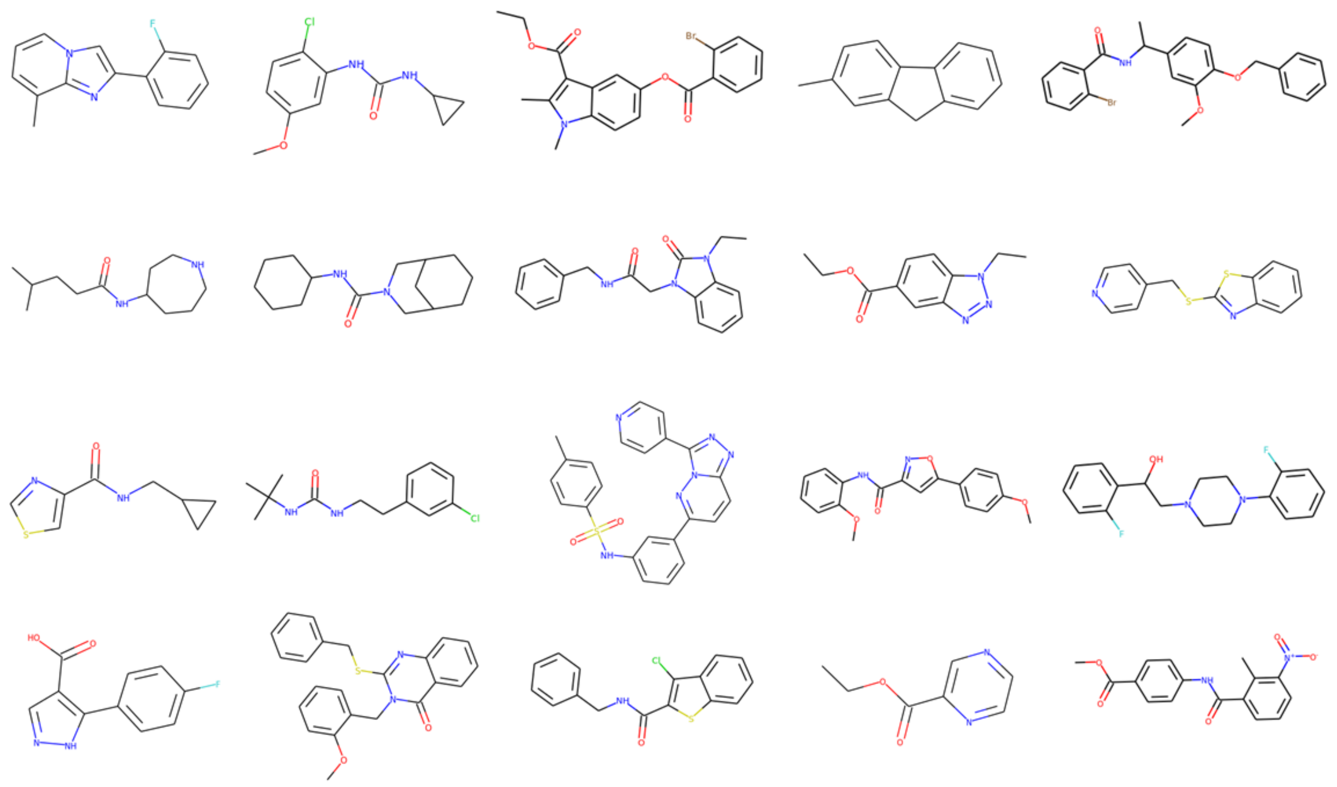

Figure 2. A sample of molecules produced by the generative model.

We have established that 95% of all generated structures, were valid, chemically-sensible molecules. The validity check was performed by the structure checker from ChemAxon (*48*). When comparing the 1M generated molecules with those from ChEMBL, the model produced less than 0.1% of structures from the training dataset. Additional comparison with the ZINC15 database (*49*) of 320M synthetically accessible drug-like molecules showed a match of about 3% (~32,000 molecules) of *de novo* generated structures could be found in ZINC. All ZINC IDs are available in Supplementary Information. These results demonstrate that our approach does produce realistic molecules while enabling the generation of predominantly novel molecules, which is the chief objective of *de novo* chemical design.

In order to characterize the structural novelty of the de novo generated molecules we compared the content of the Murcko scaffolds (*50*) between the ChEMBL training set and the virtual library generated by our system. Murcko scaffolds provide a hierarchical molecular organization scheme by dividing small molecules into R-groups, linkers, and frameworks, or scaffolds. They define the ring systems of a molecule by removing side chain atoms. We found that less than 10% of scaffolds in our library were present in ChEMBL. Overall, this analysis suggests that generative Stack RNN model did not simply memorize the training SMILEs sequences but was indeed capable of generating extremely diverse yet realistic molecules.

We assess the synthetic accessibility of generated molecules by using synthetic accessibility score (SAS).(*51*) SAS employs the knowledge extracted from known synthesis reactions with penalty for high molecular complexity. For ease of interpretability SAS scaled to be between 1 and 10. Molecules with the high SAS, typically above 6 are difficult to synthesize, whereas, molecules with the low SAS values are easily synthetically accessible. The distribution of SAS values calculated for 1M molecules generated by ReLeaSE is shown in Supplementary Figure S1. To illustrate the robustness of the approach we compared it with distribution of SAS values for full ChEMBL library (~1.5M molecules) and 1M random subsample from ZINC. Similarly, to typical commercial vendor libraries, distribution of SAS for ReLeaSE is right-skewed towards more easily synthesizable molecules. Median values are 2.9 for ChEMBL and 3.1 for both ZINC and ReLeaSE. Over 99.5% of generated molecules are below SAS of 6. Therefore, despite their high novelty, vast majority of generated compounds can be considered as synthetically accessible.

Property prediction. Over past 40 years, well defined QSAR protocols and procedures have been established (*52*, *53*). Any QSAR method can be generally defined as an application of machine learning and/or statistical methods to the problem of finding empirical relationships of the form $y = f(X_1, X_2, \ldots, X_n)$, where y is biological activity (or any property of interest) of molecules; $X_1, X_2, \ldots, X_n$ are calculated molecular descriptors of compounds; and, f is some empirically established mathematical transformation that should be applied to descriptors to calculate the property values for all molecules.

Building machine learning (ML) models directly from SMILES strings completely bypasses the most traditional QSAR step of descriptor generation. In addition to being relatively slow, descriptor generation is non-differentiable and it does not allow a straightforward inverse mapping from the descriptor space back to molecules. Albeit, a few approaches in this direction have been proposed (*54–56*). In contrast, using neural networks directly on SMILES is fully differentiable, and it enables direct mapping of properties to the SMILES sequence of characters (or strings). SMILES strings were used for model building previously (*57–59*); however, in most cases SMILES strings were used to derive string- and substring-based numerical descriptors(*60*) . Note that, in our case, the ability to develop property or activity QSAR models using SMILES was critical for integrating models possessing the *de novo* structure generation step as described below.

In terms of accuracy, SMILES based ML models also perform very well. For example, using five-fold cross validation we obtained the external model accuracy expressed as R^2_{ext} of 0.91 and RMSE = 0.53 for LogP (See Methods section). This compared favorably to a Random Forest model with DRAGON7 descriptors (R^2_{ext} = 0.90 and RMSE = 0.57). For the melting temperature prediction, the observed RMSE of 33 ^oC is on par with state-of-the-art model obtained by stacking of multiple descriptors-based ML models together (*61*), which afforded RMSE of 33 ^oC.

Generation of property value biased libraries with the RL system. To explore the utility of the RL algorithm in a drug design setting, we have conducted case studies to design libraries with three controlled target properties: a) physical properties considered important for drug-like molecules, b) specific biological activity, and c) chemical complexity. For physical properties we selected melting temperature (T_m) and n-octanol/ water partition coefficient (LogP). For bioactivity prediction, we designed putative inhibitors of Janus protein kinase 2 (JAK2) with novel chemotypes. Finally, the number of benzene rings and the number of substituents (like –OH, $-NH_2$, $-CH_3$ –CN, etc.) was used as a structural reward to design novel chemically complex compounds. Figure 3 shows the distribution of predicted properties of interest in the training test molecules and in the libraries designed by our system. In both cases, we sampled 10,000 molecules by the baseline (no RL) generator and RL-optimized generative models, and then calculated their properties with a corresponding predictive model. Values of the substructure features were calculated directly from the 2D structure. Table 1 summarizes the analysis of generated molecules and the respective statistics.

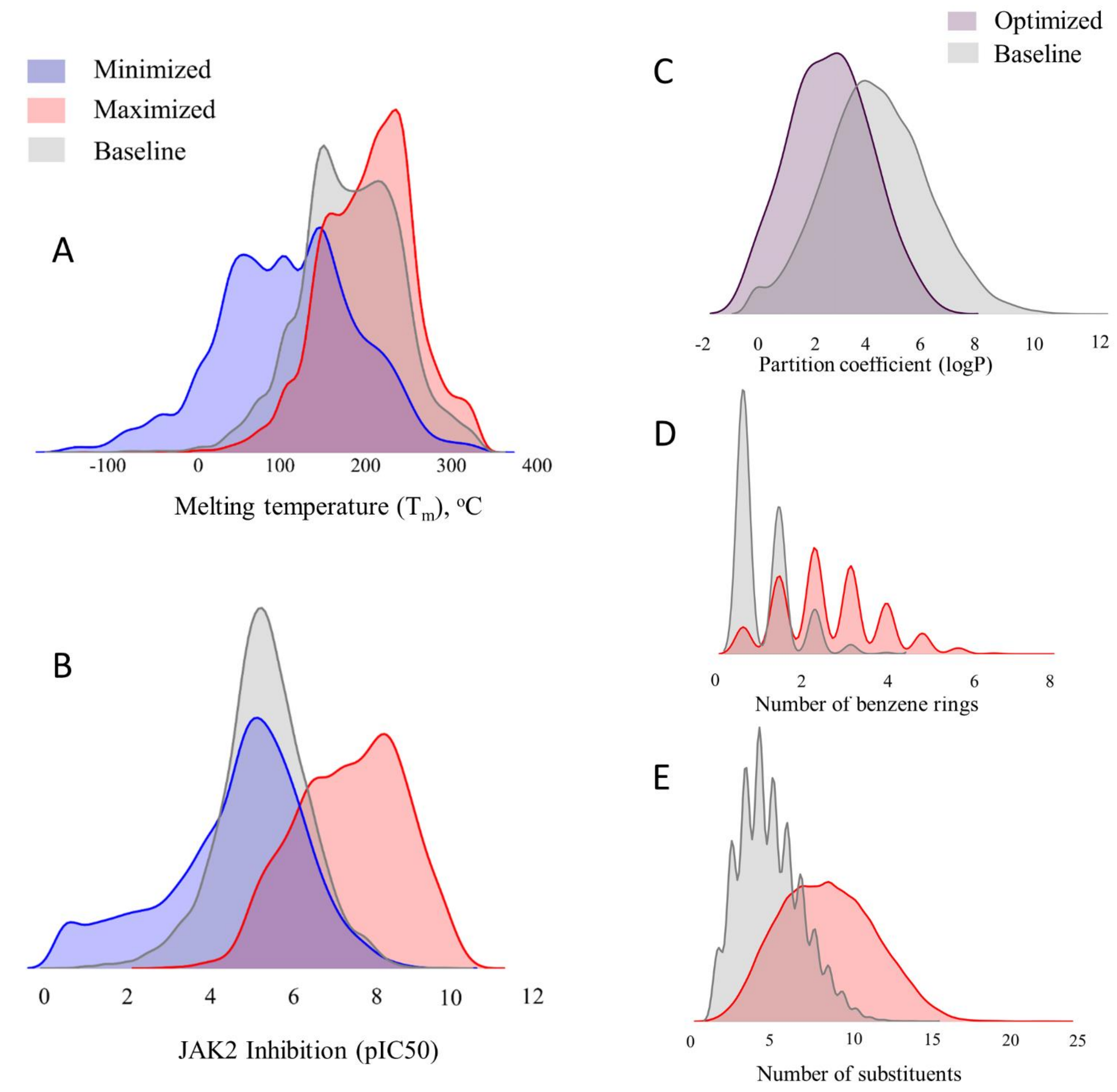

Figure 3. Property distributions for RL optimized versus baseline generator model. (A) Melting temperature **(B)** JAK2 inhibition **(C)** Partition coefficient **(D)** Number of benzene rings **(E)** Number of substituents.

Melting temperature (T_m). In this experiment, we set two goals, i.e., either to minimize or to maximize the target property. Upon minimization, the mean of the distribution in the *de novo* generated library was shifted by 44 °C as compared to the training set distribution (Figure 3A). The library of virtually synthesized chemicals included simple hydrocarbons like butane, as well as poly-halogenated compounds like CF_2Cl_2 and $C_6H_4F_2$. The molecule with the lowest T_m=-184 °C in the produced dataset was CF_4.

Clearly, this property minimization strategy was extremely effective, as it allowed for the discovery of molecules in the regions of the chemical space far beyond those of the training set of drug-like compounds. In the maximization regime, the mean of the melting temperature was increased by 20 °C to 200 °C. The generated library included substantially more complex molecules with the abundance of sulphur-containing heterocycles, phosphates, and conjugated double bond moieties.

Designing a chemical library biased toward a range of lipophilicity (LogP). Compound hydrophobicity is an important consideration in drug design. One of the components of the famous Lipinski's rule of five is that orally bioavailable compounds should have their octanol-water partition coefficient LogP less than 5 (*62*). Thus, we endeavored to design a library that would contain compounds with LogP values within a favorable drug-like range. The reward function in this case was defined as a piecewise linear function of LogP with a constant region from 1.0 to 4.0 (see Supplementary Figure S2). In other words, we set the goal to generate molecules according to a typical Lipinski's constraints. As is shown in Figure 3C, we have succeeded in generating a library with 88% of the molecules falling within the LogP drug-like region.

Inhibition of JAK2. In the third experiment, which serves as an example of the most common application of computational modeling in drug discovery, we have employed our system to design molecules with the specific biological function, i.e. JAK2 activity modulation. Specifically, we designed libraries with the goal of minimizing or maximizing pIC_{50} values for JAK2. Bioactivity minimization is also pursued in drug discovery to mitigate off-target effects. Therefore, we were interested in exploring the ability of our system to bias the design of novel molecular structures toward any desired range of the target properties. JAK2 is non-receptor tyrosine kinase involved in various processes such as cell growth, development, differentiation or histone modifications. It mediates essential signaling events in both innate and adaptive immunity. In the cytoplasm it also plays an important role in signal transduction (*63*). Mutations in JAK2 have been implicated in multiple conditions like thrombocythemia, myelofibrosis or myeloproliferative disorders (*64*).

The reward functions in both cases (min and max) were defined as exponential functions of pIC_{50} (see Supplementary Figure S2). The results of library optimization are shown in Figure 3C. With minimization, the mean of predicted pIC_{50} distribution was shifted by about one pIC_{50} unit and the distribution was heavily biased toward the lower ranges of bioactivity with 24% of molecules predicted to have practically no activity ($pIC_{50} \leq 4$). In the activity maximization exercise, properties of generated molecules were more tightly distributed across the predicted activity range. In each case, our system virtually synthesized both known and novel compounds, with the majority of de novo designs being novel molecules. The generation of known compounds (i.e. not included in the training set) can be regarded as model validation. Indeed, the system retrospectively discovered 793 commercially available compounds deposited in the ZINC database, which constituted about 5% of the total generated library. Some of them, like ZINC263823677 and ZINC271402431 were actually annotated as possible tyrosine kinase inhibitors.

Substructure bias. Finally, we also performed two simple experiments mimicking the strategy of biased chemical library design where the designed library is enriched with certain user-defined substructures. We defined the reward function as the exponent of (i) the number of benzene rings (-Ph) and (ii) total number of small groups substituents. Among all case studies described, structure bias was found to be the easiest to optimize. Figure 4 illustrates the evolution of generated structures as the structural reward increases. Indeed, we see that the model progresses toward generating increasingly more complex, yet realistic molecules with greater numbers of rings and/or substituents.

We expect that designing structurally biased libraries may be a highly desirable application of our approach as researchers often wish to generate libraries enriched for certain privileged scaffold(-s) (*65*).

Conversely, the system also allows to avoid particular chemical groups or substructures (like bromine or carboxyl group) that may lead to undesired compound properties such as toxicity. Finally, one could implement certain substructure or pharmacophore similarity (*66*) reward to explore additional chemical space.

Table 1 shows a decrease in the proportion of the valid molecules after the optimization. We may explain this phenomenon by the weaknesses of predictive models *P* (See Figure 1C) and the integration of predictive and generative models into a single design system. We presume that the generative model *G* tends to find some local optima of the reward function that correspond to invalid molecules, but predictive model *P* assigns high rewards to these molecules. This explanation is also supported by the results of structure bias optimization experiments, as we did not use any predictive models in these experiments and the decrease in the proportion of valid molecules was insignificant. We also noticed, that among all experiments with predictive models those with LogP optimization showed the highest proportion of valid molecules and, at the same time, the predictive model for LogP estimation had the highest accuracy $R^2 = 0.91$ (see Methods).

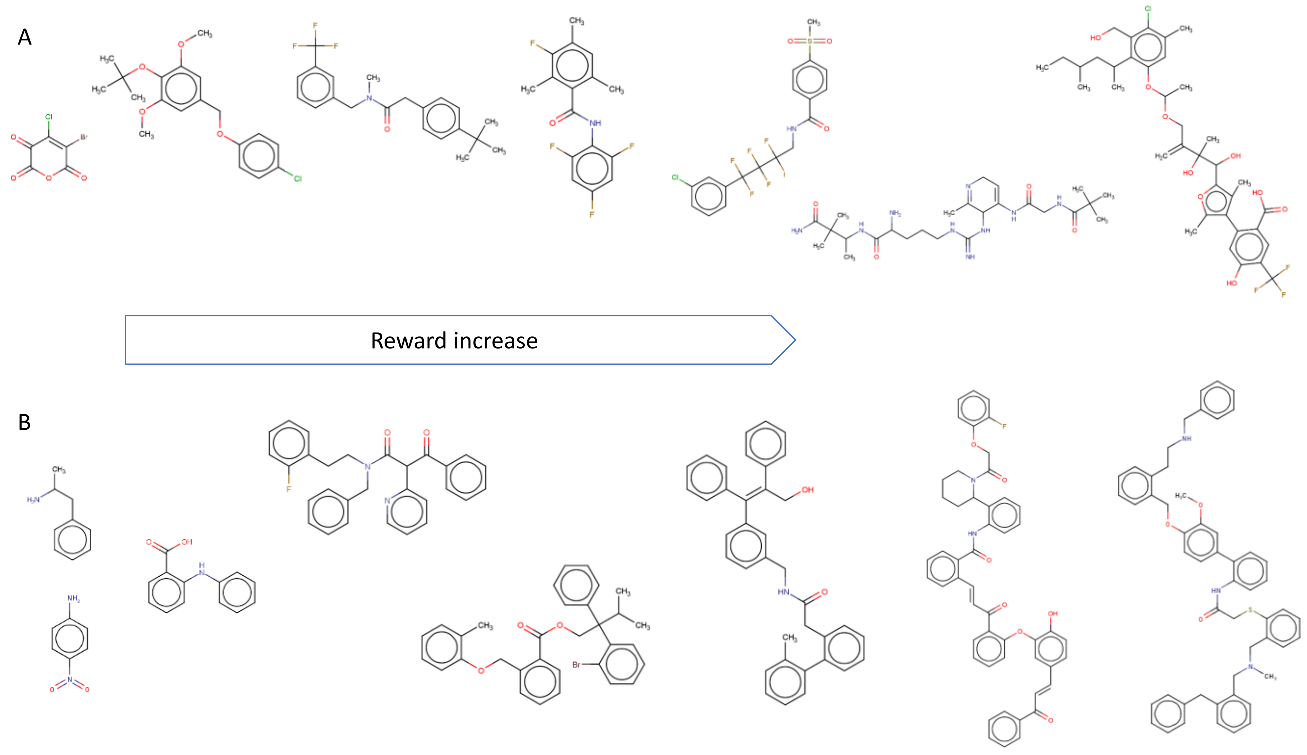

Figure 4. Evolution of generated structures as chemical substructure reward increases. (**A**) Reward proportional to the total number of small group substituents (**B**) Reward proportional to the number of benzene rings.

Table 1. Comparison of statistics for generated molecular datasets

Property		Valid molecules, %	Mean SA score	Mean molar mass	Mean value of target property	Match with ZINC15 database, %	Match with ChEMBL database, %
Melting temperature	baseline	95	3.1	435.4	181	4.7	1.5
	minimized	31	3.1	279.6	137	4.6	1.6
	maximized	53	3.4	413.2	200	2.4	0.9
pIC_{50} for JAK2	baseline	95	3.1	435.4	5.70	4.7	1.5
	minimized	60	3.85	481.8	4.89	2.5	1.0
	maximized	45	3.7	275.4	7.85	4.5	1.8
log P	baseline	95	3.1	435.4	3.63	4.7	1.5
	range opt.	70	3.2	369.7	2.58	5.8	1.8
Number of benzene rings	baseline	95	3.1	435.4	0.59	4.7	1.5
	maximized	83	3.15	496.0	2.41	5.5	1.6
Number of substituents	baseline	95	3.1	435.4	3.8	4.7	1.5
	maximized	80	3.5	471.7	7.93	3.1	0.7

Model analysis. Model interpretation is a highly significant component in any ML study. In this section we demonstrate how Stack-RNN learns and memorizes useful information from the SMILES string that it is currently being processing. We have manually analyzed neuron gate activations of the neural network as it processes the input data.

Figure 5 lists several examples of cells in neural networks with interpretable gate activations. In this figure each line corresponds to activations of a specific neuron at different processing SMILES time steps by the pre-trained baseline generative model. Each letter is colored according to the value of *tanh* activation in a cool-warm colormap from dark blue to dark red, i.e., from -1 to 1. We found that our RNN has several interpretable cells. These cells can be divided into two groups – a chemically sensible group, which activates on specific chemical groups or moieties, and a syntactic group, which keep tracks of numbers, bracket opening and closure, and even SMILES string termination of the when the new molecule is generated. For instance, we saw cells reflecting the presence of a carbonyl group, aromatic groups or NH moieties in heterocycles. We also observed that in two of these three examples there were counter-cells that deactivate in the presence of the aforementioned chemical groups. Neural network-based models are notoriously uninterpretable (*67*, *68*) and the majority of cells were indeed in that category. On the other hand, the possibility of even partial interpretation offered by this approach could be highly valuable to a medicinal chemist.

Visualization of new chemical libraries. In order to understand how the generative models populate chemical space with new molecules, we used t-Distributed Stochastic Neighbor Embedding (t-SNE) for dimensionality reduction (*69*). We selected datasets for three endpoints used in our case studies (T_m, LogP, JAK2) that were produced with corresponding optimized generative models *G*. For every molecule we calculated a latent vector of representation from the feed-forward layer with ReLU activation function in the predictive model *P* for the respective property and constructed 2D projection using t-SNE. These projections are illustrated in Figure 6. Every point corresponds to a molecule and is colored according to its property value.

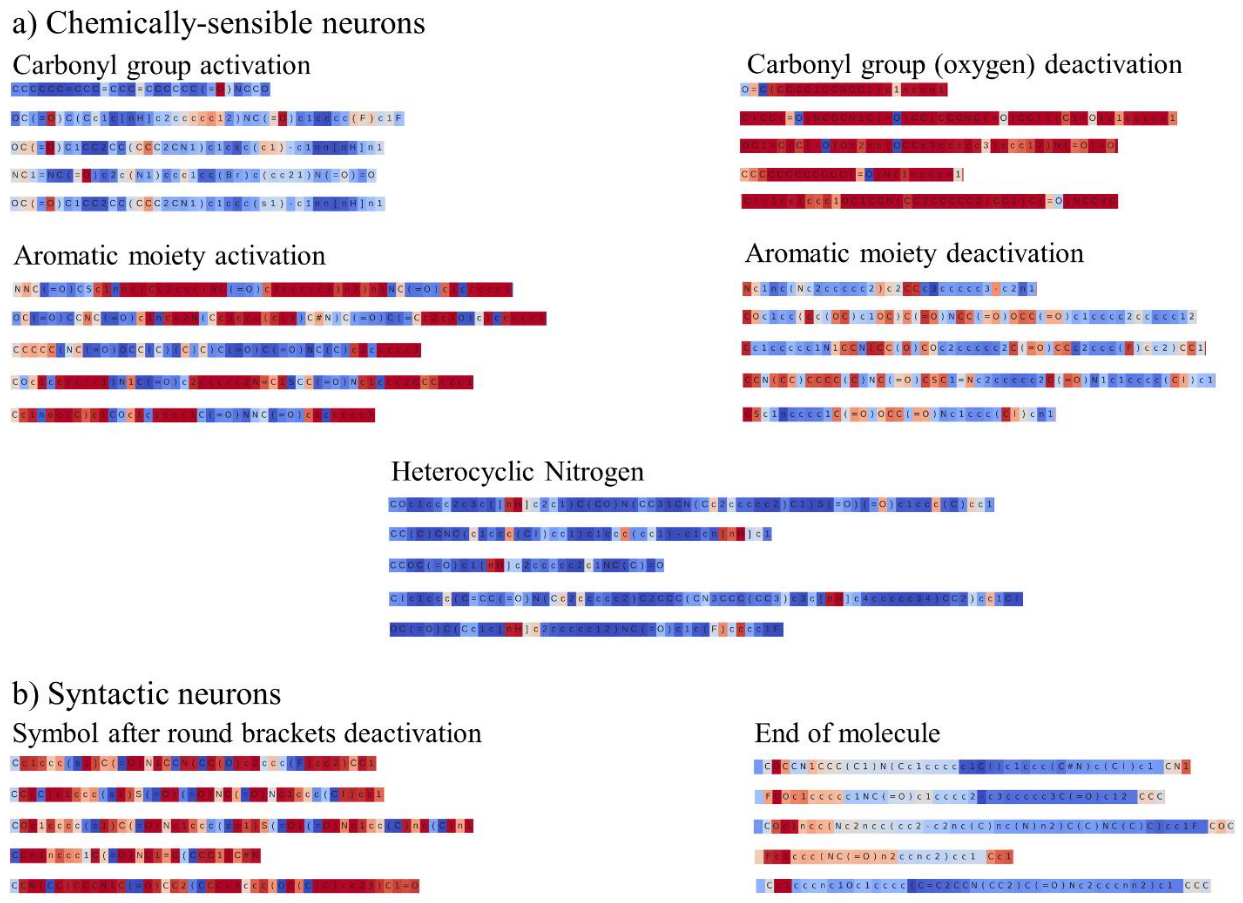
a) Chemically-sensible neurons
Carbonyl group activation
Carbonyl group (oxygen) deactivation
Aromatic moiety activation
Aromatic moiety deactivation
Heterocyclic Nitrogen
b) Syntactic neurons
Symbol after round brackets deactivation
End of molecule

Figure 5. Examples of Stack-RNN cells with interpretable gate activations. Color coding corresponds to GRU cells with hyperbolic tangent *tanh* activation function, where dark blue corresponds to the activation function value of -1 is and red described the value of the activation function of 1; the numbers in the range between -1 and 1 are colored using cool-warm color map.

For libraries generated to achieve certain partition coefficient distribution (Figure 6a), we can observe well-defined clustering of molecules with similar LogP values. In contrast, for melting temperature (Figure 6b) there are no such clusters. High and low T_m molecules are intermixed together. This observation can be explained by the fact, that melting temperature depends not only on the chemical structure of the molecule, but also on intermolecular forces as well as packing in the crystal lattice. Therefore, plotting molecules in this neural net representation could not achieve good separation of high vs. low T_m. In the case of the JAK2 model, we could observe two large non-overlapping areas roughly corresponding to inactive ($pIC_{50}<6$) and active($pIC_{50}\geq6$) compounds. Inside areas, molecules are typically clustered around multiple privileged scaffolds. Specifically for JAK2 we see abundance of compounds with 1,3,5-triazine, 1,2,4-triazine, 5-Methyl-1H-1,2,4-triazole, 7H-pyrrolo[2,3-d]pyrimidine, 1H-pyrazolo[3,4-d]pyrimidine, thieno-triazolo-pyrimidine and other substructures. Overall this approach offers a rapid way to visualize a chemical space in coordinates that are most relevant to a specific prediction endpoint. Joint embedding of training and newly generated molecules could also help to understand what parts of novel chemical space the model has already explored.

Discussion

We have implemented a deep reinforcement learning approach for *de novo* molecular design. This strategy allows the generation of novel chemical compounds with desired properties. Two deep neural networks – generative and predictive are combined in a general RL workflow. The training process consists of two stages. In the first stage, both models are trained separately using supervised learning, and in the second stage, models are trained jointly with a reinforcement learning method. Both neural networks employ end-to-end deep learning that do not rely on pre-defined chemical descriptors and are trained directly from chemical structures represented by SMILES strings. This distinction makes this approach clearly differentiated from traditional QSAR methods and simpler both to use and execute. Reinforcement Learning approach for de-novo molecular design was employed in (*70*) as well. However, no data was provided to show that the predicted properties of molecular compounds are optimized. Instead of demonstrating the shift in distribution of biological activity values against dopamine receptor type 2 (DRD2) before and after optimization, paper shows increase in fraction of generated molecules, that are similar to train and test sets. This increase doesn't mean, that the generative model is capable to produce novel active compounds, but may results from the model's weaknesses such as memorizing the training set. Indeed, the generative model is a "vanilla" recurrent neural network without augmented memory stack, which has no capacity to count and infer algorithmic patterns. (*40*) Another weakness, from our point of view, is usage of standard QSAR predictive model depending on numerical descriptors, whereas we propose a model which is essentially descriptor-free and naturally forms a coherent workflow together with the generative model.

As a proof of principle, we tested our approach on three diverse types of endpoints: physical properties, biological activity and chemical substructure bias. The use of flexible reward function enables different library optimization strategies where one can minimize, maximize or impose a desired range to a property of interest in the generated compound libraries. As a by-product of these case studies, we have generated a dataset of over 1M of novel compounds that our model synthesized virtually.

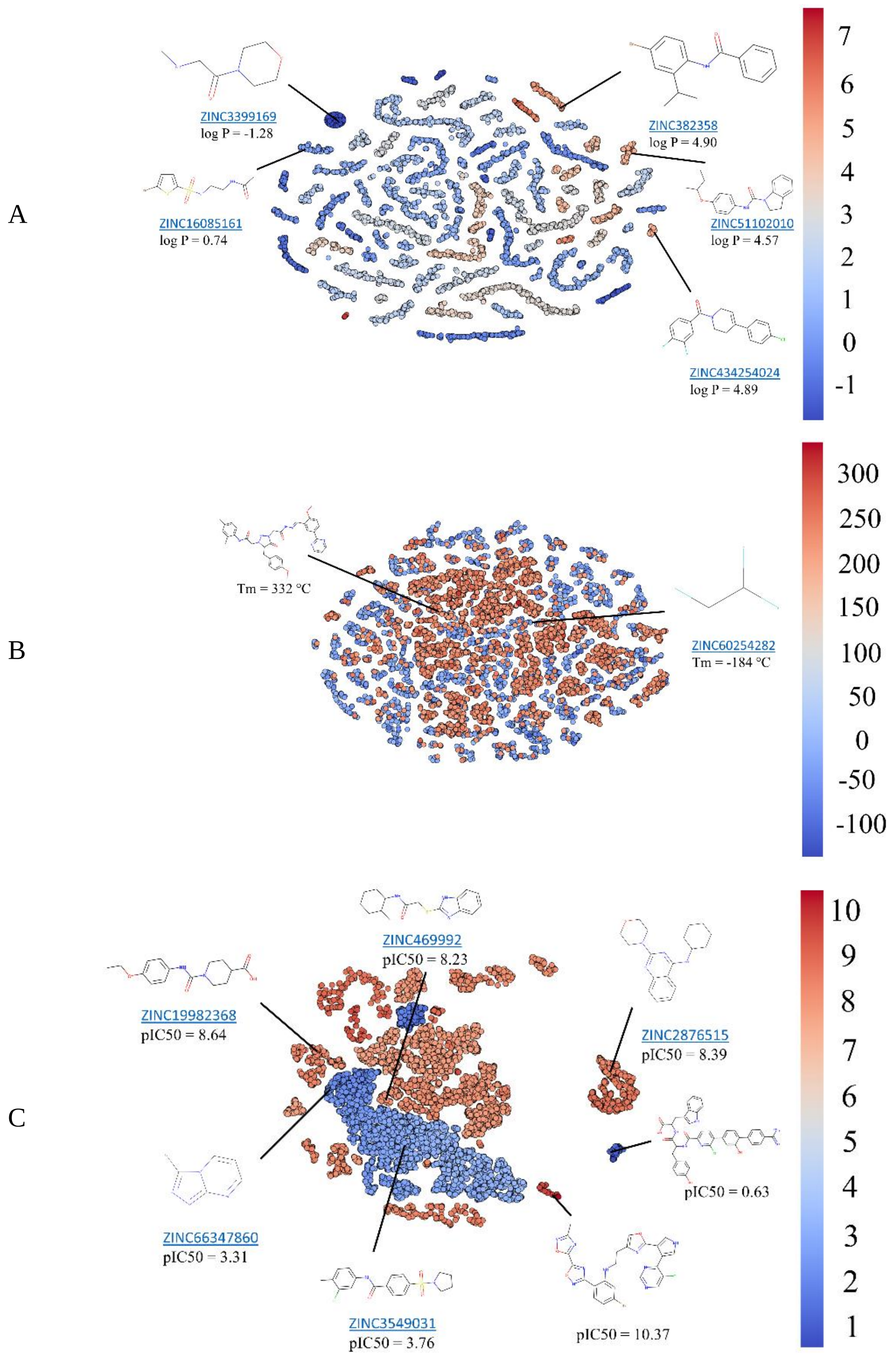

Figure 6. Clustering of generated molecules by t-distributed stochastic neighbor embedding (t-SNE). Molecules are colored based on the predicted properties by model *P*, with values shown by the color bar on the right. **(A, C)** Examples of generated molecules are randomly picked from matches with ZINC database, property values predicted by model *P* **(A)** Partition coefficient, logP **(B)** Melting temperature, T_m ^oC, examples show generated molecules with lowest and highest predicted melting temperatures **(C)** JAK2 inhibition, predicted pIC_{50}.

The common critique for methods of computational library design is often their inability to control synthetic accessibility of produced molecules. Indeed, computationally generated compounds are often quite complex with exotic substituents. In many cases require multi-step custom syntheses or even could not be synthesized with current level of technology. In the pharmaceutical industry, the ease of synthesis of a prospective hit molecule is of primary concern as it strongly affects the cost of the manufacturing process required for industrial-scale production. For all experiments in this paper, synthetic accessibility of generated focus libraries was estimated using SAS score.(*51*) Distributions of SAS values are depicted in Supplementary Figure S3, medians are listed in Table 1. It is clearly seen, that property optimization does not significantly affect synthetic accessibility of produced molecules. The biggest shift of 0.75 for the distribution median was observed for minimization of JAK2 inhibition. Less than 0.5% of molecules had a high SAS of >6, that an approximate cutoff for systems that are difficult to synthesize.

In our opinion, there are two main reasons, why it is not feasible to include SAS, at least in its current form, in the reward function. First, our ReLeaSE model is very stable: SAS is practically independent from property optimization, their distribution follows one from a commercially available compound space. Second, “synthetic accessibility” is not a well-defined concept.(*71*) In process chemistry it is refers to the numerous factors that determine the ease of a synthesis of a particular molecule. These include the availability of reagents, the number and difficulty of synthetic steps, the stability of intermediate products, ease of their separation, reaction yields etc.(*72*) However no such global score is available. In contrast, the method of choice (also used in this work) is based on molecular complexity as defined by a number of substructures and molecular fragments.(*51*) Therefore, its direct optimization with RL will result in substantially reduced novelty of generated molecules and a bias toward substructures with low SAS scores that used to train a model. Overall, the lack of a rigorous definition for SA also makes its determination and comparison a non-trivial task. Despite being very active area of research, very little progress has so far been achieved for past 10 years.

In summary, we have devised and executed a new strategy for designing libraries of compounds with the desired properties that employs both deep learning and reinforcement learning approaches. We have termed our method ReLeaSE for Reinforcement Learning for Structural Evolution (mindful that one of the key meanings of the word “release” is to “allow or enable to escape from confinement; set free”). We have conducted computational experiments that demonstrated the efficiency of the proposed ReLeaSE strategy in a single-task regime where each of the endpoints of interest is optimized independently. However, this system can be extended to afford multi-objective optimization of several target properties concurrently, which is the need of drug discovery where the drug molecule should be optimized with respect to potency, selectivity, solubility, and ADMET properties. Our future studies will address this challenge.

Materials and Methods

Data. The melting point dataset was extracted from the literature (*61*). The PHYSPROP database (*73*) used to extract the octanol/water partition coefficient, LogP for diverse set of molecules. Experimental IC_{50} and K_i data tested against JAK2 (CHEMBL ID 2971) was extracted from ChEMBL (*47*), PubChem (*74*) and Eidogen-Sertanty KKB (*75*). Compounds that had inconclusive IC_{50} values were considered unreliable and were not included in the modeling.

Data curation. Compiled datasets of compounds were carefully curated following the protocols proposed by Fourches et al. (*76*) Briefly, explicit hydrogens were added, and specifics chemotypes such as aromatic and nitro groups were normalized using ChemAxon Standardizer. Polymers, inorganic salts, organometallic compounds, mixtures, and duplicates were removed. Modeling-ready curated dataset contained 14,176 compounds for LogP, 15,549 compounds for JAK2 and 47,425 for melting temperature. All molecules were stored as normalized and canonicalized SMILES strings according to procedures developed elsewhere (*77*).

Predictive models. We trained predictive models for three different properties – melting temperature, LogP and pIC_{50} for JAK2. Each model consists of embedding layer, which transforms sequence of discrete tokens into a vector of 100 continuous numbers, LSTM layer with 100 units and *tanh* nonlinearity, one dense layers with 100 units and rectify nonlinearity function and one dense layer with one unit and identity activation function. All three models were trained with learning rate decay technique until convergence. Curated datasets were divided into training and validation sets in 3:1 ratio. The results and accuracy of the model are shown in Figure S4.

Training. In the first stage, we pre-train a generative model on a ChEMBL21 (*47*) dataset of approximately 1.5M drug-like compounds, so that the model is capable of producing chemically feasible molecules, but without property optimization. This network has 1500 units in a recurrent GRU (*42*) layer and 512 units in a stack augmentation layer. The model was trained on a GPU for 10000 epochs. The learning curve is illustrated in Figure S5.

The generative model has two modes of processing sequences – training and generating. At each time step, in training mode, the generative network takes a current prefix of the training object and predicts the probability distribution of the next character. Then, the next character is sampled from this predicted probability distribution and is compared to the ground truth. Afterwards, based on this comparison the cross-entropy loss function is calculated and parameters of the model are updated. At each time step, in generating mode, the generative network takes a prefix of already generated sequences and then, like in the training mode, predicts the probability distribution of the next character and samples it from this predicted distribution. In the generative model, we do not update the model parameters.

At the second stage, we combine both generative and predictive model into one reinforcement learning system. In this system, the generative model plays the role of an agent, whose action space is represented by the SMILES notation alphabet and state space is represented by all possible strings in this alphabet. The predictive model plays the role of a critic estimating the agent's behavior by assigning a numerical reward to every generated molecule (i.e., SMILES string). The reward is a function of the numerical property calculated by the predictive model. At this stage, the generative model is trained to maximize the expected reward. The whole pipeline is illustrated in Figure 1.

We trained a stack-augmented RNN as a generative model. As a training dataset we used the ChEMBL database of drug-like compounds, which includes approximately 1.5 million of SMILES strings. For training we selected just those molecules from the initial training dataset whose SMILES notation lengths were fewer than 100 characters. The length of 100 was chosen because more than 97% of SMILES in training dataset had 100 characters or fewer (see Figure S6).

Stack-augmented recurrent neural network (*40*)**.** This section describes generative model *G* in more details. We assume that the data is sequential, which means that it comes in the form of discrete tokens, i.e. characters. The goal is to build a model, that is able to predict the next token conditioning on all previous tokens. The regular recurrent neural network has an input layer and a hidden layer. At time step *t* neural network takes the embedding vector of token number t from the sequence as an input and models the probability distribution of the next token given all previous tokens, so that the next token can be sampled from this distribution. Information of all previously observed tokens is aggregated in the hidden layer. This can be written down as the following:

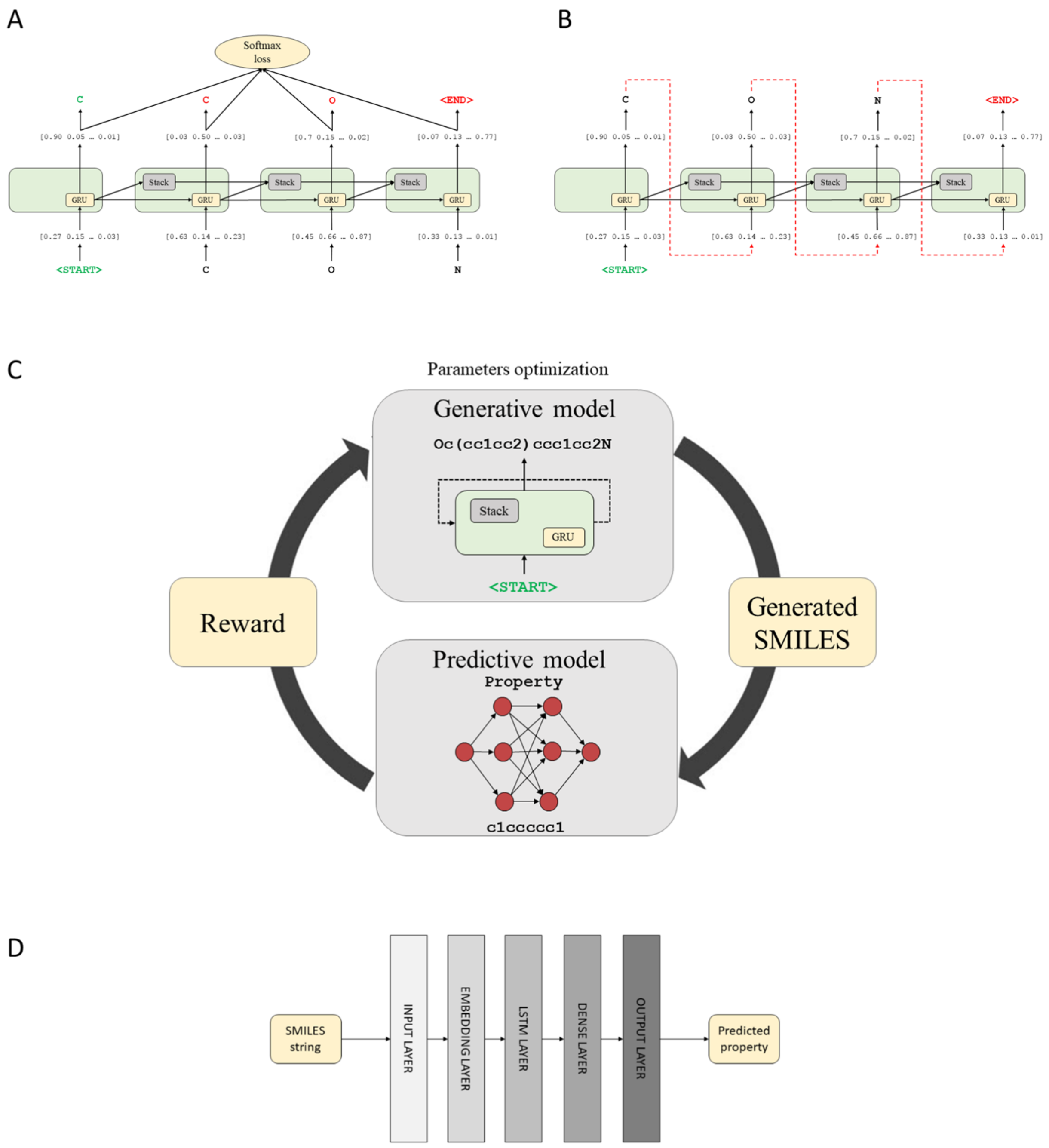

Figure 1. The Deep RL algorithm for generating new SMILES strings of compounds with the desired properties. A. Training step of the generative stack-augmented RNN; **B**. Generator step of the generative stack-augmented RNN. During training, the input token is a character in the currently processed SMILES string from the training set. The model outputs probability vector $p_\Theta(a_t|s_{t-1})$ of the next character given a prefix. Vector of parameters Θ is optimized by cross-entropy loss function minimization. In the generator regime, the input token is a previously generated character. Next, character a_t is sampled randomly from the distribution $p_\Theta(a_t|s_{t-1})$. **C.** General pipeline of reinforcement learning system for novel compounds generation. **D.** Scheme of predictive model. This model takes a SMILES string as an input and provides one real number, which is an estimated property value, as an output. Parameters of the model are trained by l_2 squared loss function minimization.

$$h_t = \sigma(W_i x_t + W_h h_{t-1}), \tag{6}$$

where h_t is a vector of hidden states, h_{t-1} – vector of hidden states from the previous time step, x_t – input vector at time step *t*, W_i – parameters of the input layers, W_h -- parameter of the hidden layer and σ – activation function.

The stack memory is used to keep the information and deliver it to the hidden layer at the next time step. A stack is a type of persistent memory which can be only accessed through its topmost element. There are three basic operations supported by the stack: `POP` operation, which deletes an element from the top of the stack, `PUSH` operation, which puts a new element to the top of our stack; and also `NO−OP` operation, which performs no action. The top element of the stack has value $s_t[0]$ and is stored at position 0:

$$s_t[0] = a_t[PUSH]\sigma(Dh_t) + a_t[POP]s_{t-1}[1] + a_t[NO-OP]s_{t-1}[0]. \tag{7}$$

where D is $1 \times m$ matrix and $a_t = \left[a_t[PUSH], a_t[POP], a_t[NO-OP]\right]$ is a vector of stack control variables, which define the next operation to be performed. If $a_t[POP]$ is equal to 1, then the value below is used to replace the top element of the stack. If $a_t[PUSH]$ is equal to 1, then a new value will be added to the top and all the rest values will be moved down. If $a_t[NO-OP]$ equals 1 then stack keeps the same value on top.

Similar rule is applied to the elements of the stack at a depth i>0:

$$s_t[i] = a_t[PUSH]s_{t-1}[i-1] + a_t[POP]s_{t-1}[i+1] + a_t[NO-OP]s_{t-1}[i]. \tag{8}$$

Now the hidden layer h_t is updated as:

$$h_t = \sigma(Ux_t + Rh_{t-1} + Ds_{t-1}^k), \tag{9}$$

where D is a matrix of size $m \times k$ and s_{t-1}^k are the first *k* elements for the top of the stack at time step *t*-1.

Acknowledgments

A. T. and O. I. acknowledge support from the Eshelman Institute for Innovation award, DOD-ONR (N00014-16-1-2311). M. P. acknowledges financial support from the Skolkovo Institute of Science and Technology. O.I. acknowledges Extreme Science and Engineering Discovery Environment (XSEDE) award DMR-110088, which is supported by National Science Foundation grant number ACI-1053575. We gratefully acknowledge the support and hardware donation of NVIDIA Corporation and personally Mark Berger. We also thank Stephen Cappuzzi for technical discussions and reading the manuscript prior the publication.

Author contributions: A.T and O.I. devised and led this project. M.P. and O.I. developed and implemented method. M.P. wrote the manuscript with input from all authors.

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

Data and materials availability: All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. Code and datasets are available from GitHub (Link will be added upon manuscript acceptance). Additional data related to this paper may be requested from the authors.

References and Notes

1. Y. Gil, M. Greaves, J. Hendler, H. Hirsh, Amplify scientific discovery with artificial intelligence. *Science (80-.).* **346**, 171–172 (2014).
2. Artificial intelligence: The return of the machinery question. *Econ.*, (available at http://www.economist.com/news/special-report/21700761-after-many-false-starts-artificial-intelligence-has-taken-will-it-cause-mass).
3. C. Krittanawong, H. Zhang, Z. Wang, M. Aydar, T. Kitai, Artificial Intelligence in Precision Cardiovascular Medicine. *J. Am. Coll. Cardiol.* **69**, 2657–2664 (2017).
4. S. Jha, E. J. Topol, Adapting to Artificial Intelligence. *JAMA*. **316**, 2353 (2016).
5. K. Chockley, E. Emanuel, The End of Radiology? Three Threats to the Future Practice of Radiology. *J. Am. Coll. Radiol.* **13**, 1415–1420 (2016).
6. H. Altae-Tran, B. Ramsundar, A. S. Pappu, V. Pande, Low Data Drug Discovery with One-Shot Learning. *ACS Cent. Sci.* **3**, 283–293 (2017).
7. A. Lusci, G. Pollastri, P. Baldi, Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *J. Chem. Inf. Model.* **53**, 1563–75 (2013).
8. E. Gawehn, J. A. Hiss, G. Schneider, Deep Learning in Drug Discovery. *Mol. Inform.* **35**, 3–14 (2016).
9. M. Ragoza, J. Hochuli, E. Idrobo, J. Sunseri, D. R. Koes, Protein–Ligand Scoring with Convolutional Neural Networks. *J. Chem. Inf. Model.* **57**, 942–957 (2017).
10. A. Aliper *et al.*, Deep Learning Applications for Predicting Pharmacological Properties of Drugs and Drug Repurposing Using Transcriptomic Data. *Mol. Pharm.* **13**, 2524–2530 (2016).
11. M. H. S. Segler, M. P. Waller, Modelling Chemical Reasoning to Predict and Invent Reactions. *Chem. - A Eur. J.* **23**, 6118–6128 (2017).
12. J. S. Smith, O. Isayev, A. E. Roitberg, ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.* **8**, 3192–3203 (2017).
13. K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, A. Tkatchenko, Quantum-chemical insights from deep tensor neural networks. *Nat. Commun.* **8**, 13890 (2017).
14. A. D. Roses, Pharmacogenetics in drug discovery and development: a translational perspective. *Nat. Rev. Drug Discov.* **7**, 807–817 (2008).
15. C. A. Lipinski, Lead- and drug-like compounds: the rule-of-five revolution. *Drug Discov. Today Technol.* **1**, 337–341 (2004).
16. C. Lipinski, A. Hopkins, Navigating chemical space for biology and medicine. *Nature*. **432**, 855–61 (2004).
17. M. J. Waring *et al.*, An analysis of the attrition of drug candidates from four major pharmaceutical companies. *Nat. Rev. Drug Discov.* **14**, 475–486 (2015).
18. V. Schnecke, J. Boström, Computational chemistry-driven decision making in lead generation. *Drug Discov. Today*. **11** (2006), pp. 43–50.
19. R. Macarron, Critical review of the role of HTS in drug discovery. *Drug Discov. Today*. **11** (2006), pp. 277–279.
20. G. Schneider, U. Fechner, Computer-based de novo design of drug-like molecules. *Nat. Rev. Drug Discov.* **4**, 649–663 (2005).
21. H. Mauser, W. Guba, Recent developments in de novo design and scaffold hopping. *Curr. Opin.*

Drug Discov. Devel. **11**, 365–74 (2008).
22. P. G. Polishchuk, T. I. Madzhidov, A. Varnek, Estimation of the size of drug-like chemical space based on GDB-17 data. *J. Comput. Aided. Mol. Des.* **27**, 675–679 (2013).
23. J. Reymond, The Chemical Space Project. *Acc. Chem. Res.* **48**, 722–730 (2015).
24. J. Besnard *et al.*, Automated design of ligands to polypharmacological profiles. *Nature*. **492**, 215–20 (2012).
25. D. Reker, P. Schneider, G. Schneider, Multi-objective active machine learning rapidly improves structure–activity models and reveals new protein–protein interaction inhibitors. *Chem. Sci.* **7**, 3919–3927 (2016).
26. D. Reker, G. Schneider, Active-learning strategies in computer-assisted drug discovery. *Drug Discov. Today*. **20**, 458–465 (2015).
27. P. Schneider, G. Schneider, De Novo Design at the Edge of Chaos. *J. Med. Chem.* **59**, 4077–4086 (2016).
28. N. Brown, B. McKay, F. Gilardoni, J. Gasteiger, A Graph-Based Genetic Algorithm and Its Application to the Multiobjective Evolution of Median Molecules. *J. Chem. Inf. Comput. Sci.* **44**, 1079–1087 (2004).
29. R. Gómez-Bombarelli *et al.*, Automatic chemical design using a data-driven continuous representation of molecules. *arXiv Prepr.*, 1–23 (2016).
30. M. H. S. Segler, T. Kogej, C. Tyrchan, M. P. Waller, Generating Focussed Molecule Libraries for Drug Discovery with Recurrent Neural Networks (2017) (available at http://arxiv.org/abs/1701.01329).
31. A. Kadurin *et al.*, The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget* (2016), doi:10.18632/oncotarget.14073.
32. K. De Asis, J. F. Hernandez-Garcia, G. Z. Holland, R. S. Sutton, Multi-step Reinforcement Learning: A Unifying Algorithm. *arXiv Prepr.* (2017) (available at http://arxiv.org/abs/1703.01327).
33. M. Krakovsky, Reinforcement renaissance. *Commun. ACM*. **59**, 12–14 (2016).
34. D. Silver *et al.*, Mastering the game of Go with deep neural networks and tree search. *Nature*. **529**, 484–489 (2016).
35. H. J. van den Herik, J. W. H. M. Uiterwijk, J. van Rijswijck, Games solved: Now and in the future. *Artif. Intell.* **134**, 277–311 (2002).
36. SMILES strings. *www.opensmiles.org*, (available at www.opensmiles.org).
37. R. J. Willia, Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* **8**, 229–256 (1992).
38. J. Martens, Generating Text with Recurrent Neural Networks. *Neural Networks*. **131**, 1017–1024 (2011).
39. A. Karpathy, J. Johnson, L. Fei-Fei, Visualizing and Understanding Recurrent Networks. *arXiv Prepr.*, 1–13 (2015).
40. A. Joulin, T. Mikolov, Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets (2015) (available at http://arxiv.org/abs/1503.01007).
41. S. Hochreiter, J. Schmidhuber, Long Short-Term Memory. *Neural Comput.* **9**, 1735–1780 (1997).
42. J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*, 1–9 (2014).
43. T. Deleu, J. Dureau, Learning Operations on a Stack with Neural Turing Machines. *arXiv*, 1–6 (2016).
44. J. Berstel, *Transductions and Context-Free Languages* (Vieweg+Teubner Verlag, Wiesbaden, 1979; http://link.springer.com/10.1007/978-3-663-09367-1).

45. E. Grefenstette, K. M. Hermann, M. Suleyman, P. Blunsom, Learning to Transduce with Unbounded Memory. *arXiv Prepr.* (2015) (available at http://arxiv.org/abs/1506.02516).
46. T. Mikolov, K. Chen, G. Corrado, J. Dean, Distributed Representations of Words and Phrases and their Compositionality. *Nips*, 1–9 (2013).
47. A. P. Bento *et al.*, The ChEMBL bioactivity database: An update. *Nucleic Acids Res.* **42** (2014), doi:10.1093/nar/gkt1031.
48. ChemAxon, Marvin Sketch. *https://www.chemaxon.com/products/marvin/* (2013).
49. J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, R. G. Coleman, ZINC: A free tool to discover chemistry for biology. *J. Chem. Inf. Model.* **52** (2012), pp. 1757–1768.
50. G. W. Bemis, M. A. Murcko, The Properties of Known Drugs. 1. Molecular Frameworks. *J. Med. Chem.* **39**, 2887–2893 (1996).
51. P. Ertl, A. Schuffenhauer, Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminform.* **1**, 8 (2009).
52. A. Tropsha, Best Practices for QSAR Model Development, Validation, and Exploitation. *Mol. Inform.* **29**, 476–488 (2010).
53. A. Cherkasov *et al.*, QSAR Modeling: Where have you been? Where are you going to? *J. Med. Chem.* **57**, 4977–5010 (2014).
54. S. J. Cho, W. Zheng, A. Tropsha, Rational Combinatorial Library Design. 2. Rational Design of Targeted Combinatorial Peptide Libraries Using Chemical Similarity Probe and the Inverse QSAR Approaches. *J. Chem. Inf. Comput. Sci.* **38**, 259–268 (1998).
55. R. Brüggemann *et al.*, The Use of Hasse Diagrams as a Potential Approach for Inverse QSAR. *SAR QSAR Environ. Res.* **11**, 473–487 (2001).
56. T. Miyao, H. Kaneko, K. Funatsu, Inverse QSPR/QSAR Analysis for Chemical Structure Generation (from y to x). *J. Chem. Inf. Model.* **56**, 286–299 (2016).
57. A. A. Toropov, E. Benfenati, SMILES as an alternative to the graph in QSAR modelling of bee toxicity. *Comput. Biol. Chem.* **31**, 57–60 (2007).
58. Andrey A. Toropov, Emilio Benfenati, SMILES in QSPR/QSAR Modeling: Results and Perspectives. *Curr. Drug Discov. Technol.* **4**, 77–116 (2007).
59. I. S. Haque, V. S. Pande, W. P. Walters, SIML: A Fast SIMD Algorithm for Calculating LINGO Chemical Similarities on GPUs and CPUs. *J. Chem. Inf. Model.* **50**, 560–564 (2010).
60. H. Ikebata, K. Hongo, T. Isomura, R. Maezono, R. Yoshida, Bayesian molecular design with a chemical language model. *J. Comput. Aided. Mol. Des.* **31**, 379–391 (2017).
61. I. V Tetko *et al.*, How Accurately Can We Predict the Melting Points of Drug-like Compounds? *J. Chem. Inf. Model.* **54**, 3320–3329 (2014).
62. C. A. Lipinski, F. Lombardo, B. W. Dominy, P. J. Feeney, Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Deliv. Rev.* **46**, 3–26 (2001).
63. M. Sakatsume *et al.*, The Jak kinases differentially associate with the alpha and beta (accessory factor) chains of the interferon gamma receptor to form a functional receptor unit capable of activating STAT transcription factors. *J. Biol. Chem.* **270**, 17528–34 (1995).
64. R. Kralovics *et al.*, A Gain-of-Function Mutation of JAK2 in Myeloproliferative Disorders. *N. Engl. J. Med.* **352**, 1779–1790 (2005).
65. M. E. Welsch, S. A. Snyder, B. R. Stockwell, Privileged scaffolds for library design and drug discovery. *Curr. Opin. Chem. Biol.* **14**, 347–361 (2010).
66. D. Stumpfe, J. Bajorath, Similarity searching. *Wiley Interdiscip. Rev. Comput. Mol. Sci.* **1**, 260–282 (2011).

67. C. Szegedy *et al.*, Intriguing properties of neural networks. *arXiv Prepr. arXiv …*, 1–10 (2013).
68. A. Nguyen, J. Yosinski, J. Clune, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2015), vol. 07–12–June, pp. 427–436.
69. L. J. P. Van Der Maaten, G. E. Hinton, Visualizing high-dimensional data using t-sne. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008).
70. M. Olivecrona, T. Blaschke, O. Engkvist, H. Chen, Molecular De Novo Design through Deep Reinforcement Learning (2017) (available at http://arxiv.org/abs/1704.07555).
71. M. S. Lajiness, G. M. Maggiora, V. Shanmugasundaram, Assessment of the Consistency of Medicinal Chemists in Reviewing Sets of Compounds. *J. Med. Chem.* **47**, 4891–4896 (2004).
72. T. Y. Zhang, Process Chemistry: The Science, Business, Logic, and Logistics. *Chem. Rev.* **106**, 2583–2595 (2006).
73. J. A. Beauman, P. H. Howard, Physprop database. *Syracuse Res. Syracuse, NY, USA* (1995).
74. Y. Wang *et al.*, PubChem BioAssay: 2017 update. *Nucleic Acids Res.* **45**, D955–D963 (2017).
75. Eidogen-Sertanty, (available at http://www.eidogen.com).
76. D. Fourches, E. Muratov, A. Tropsha, Trust, but verify: On the importance of chemical structure curation in cheminformatics and QSAR modeling research. *J Chem Inf Model*. **50**, 1189–1204 (2010).
77. N. M. O'Boyle, Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI. *J. Cheminform.* **4**, 22 (2012).

Supplementary Materials

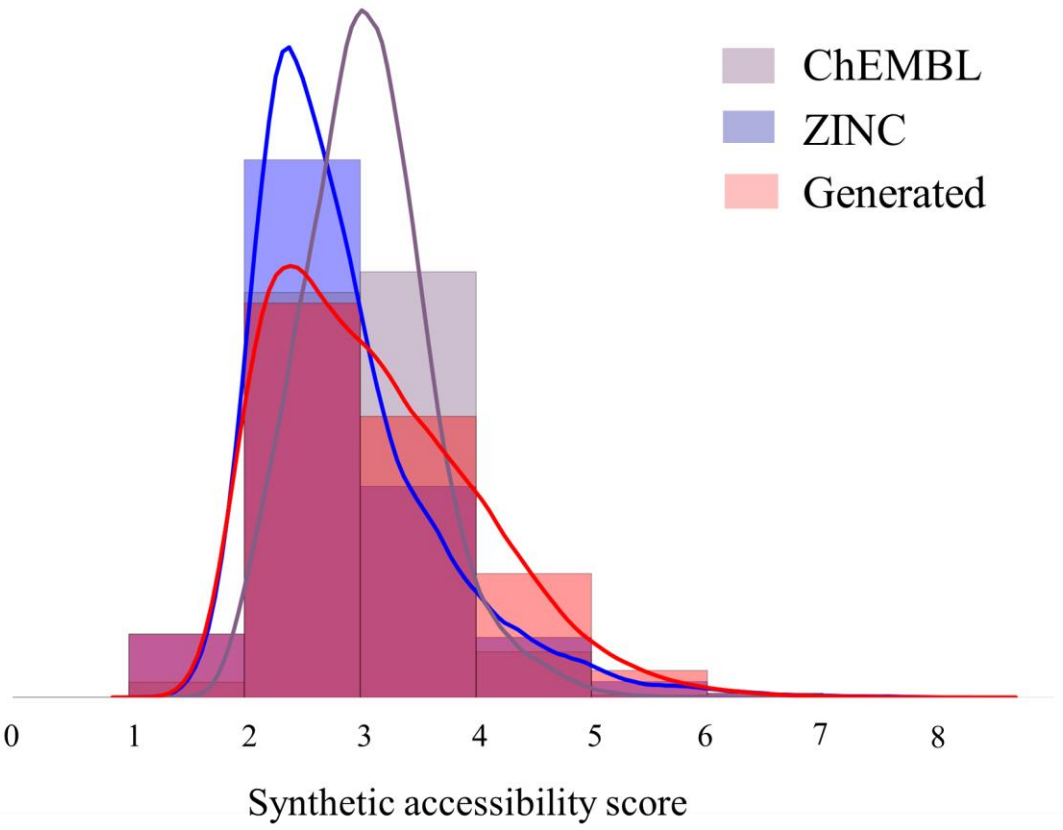

Figure S1. Distribution of Synthetic Accessibility Score (SAS) for the full ChEMBL21 database (~1.5M molecules), random subsample of 1M molecules from ZINC15 and generated dataset of 1M molecules with baseline generator model G.

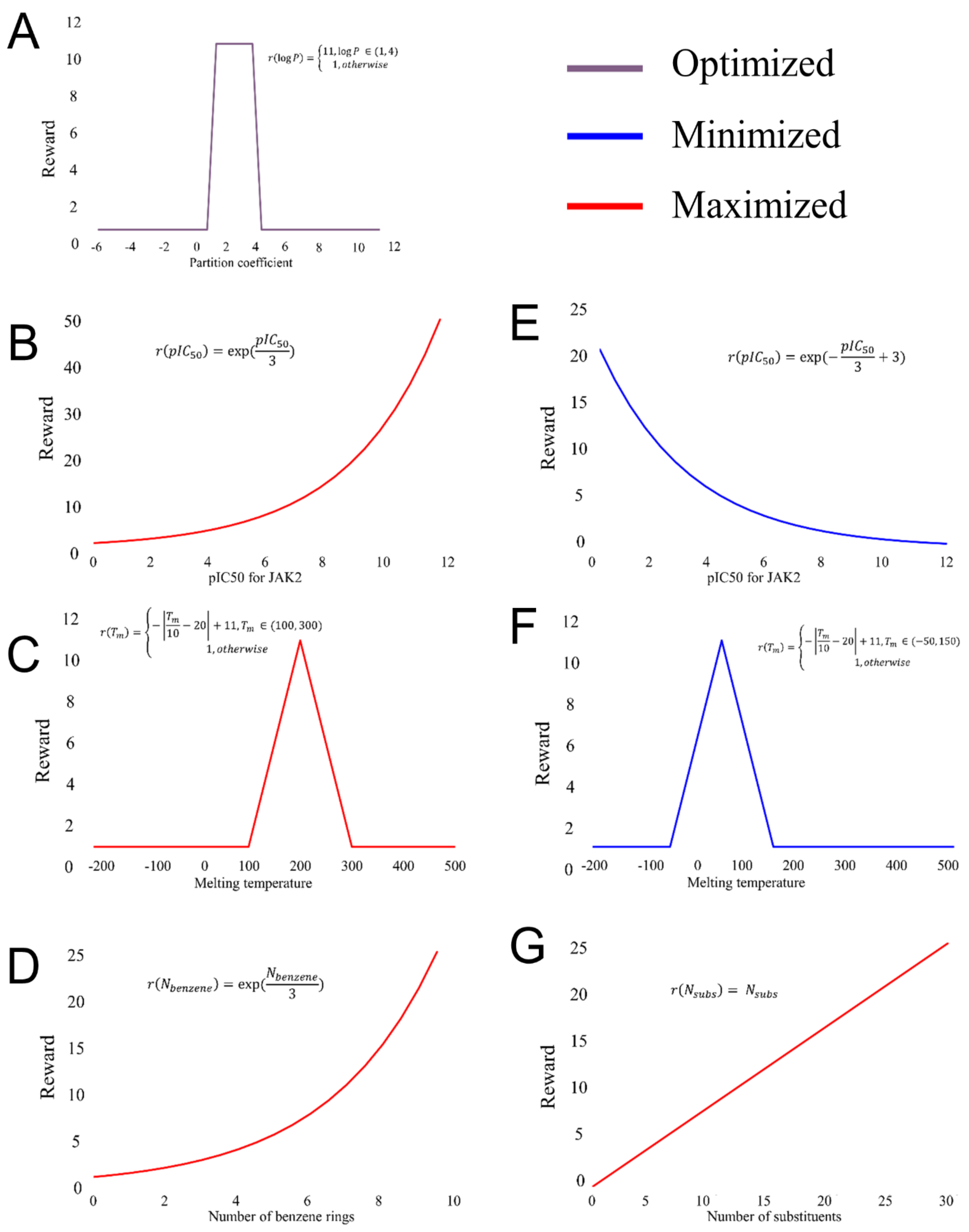

Figure S2. Reward functions. (A) logP optimization **(B)** pIC_{50} for JAK2 maximization **(C)** Melting temperature maximization **(D)** Benzene rings maximization **(E)** pIC_{50} for JAK2 minimization **(F)** Melting temperature minimization **(G)** Substituents maximization.

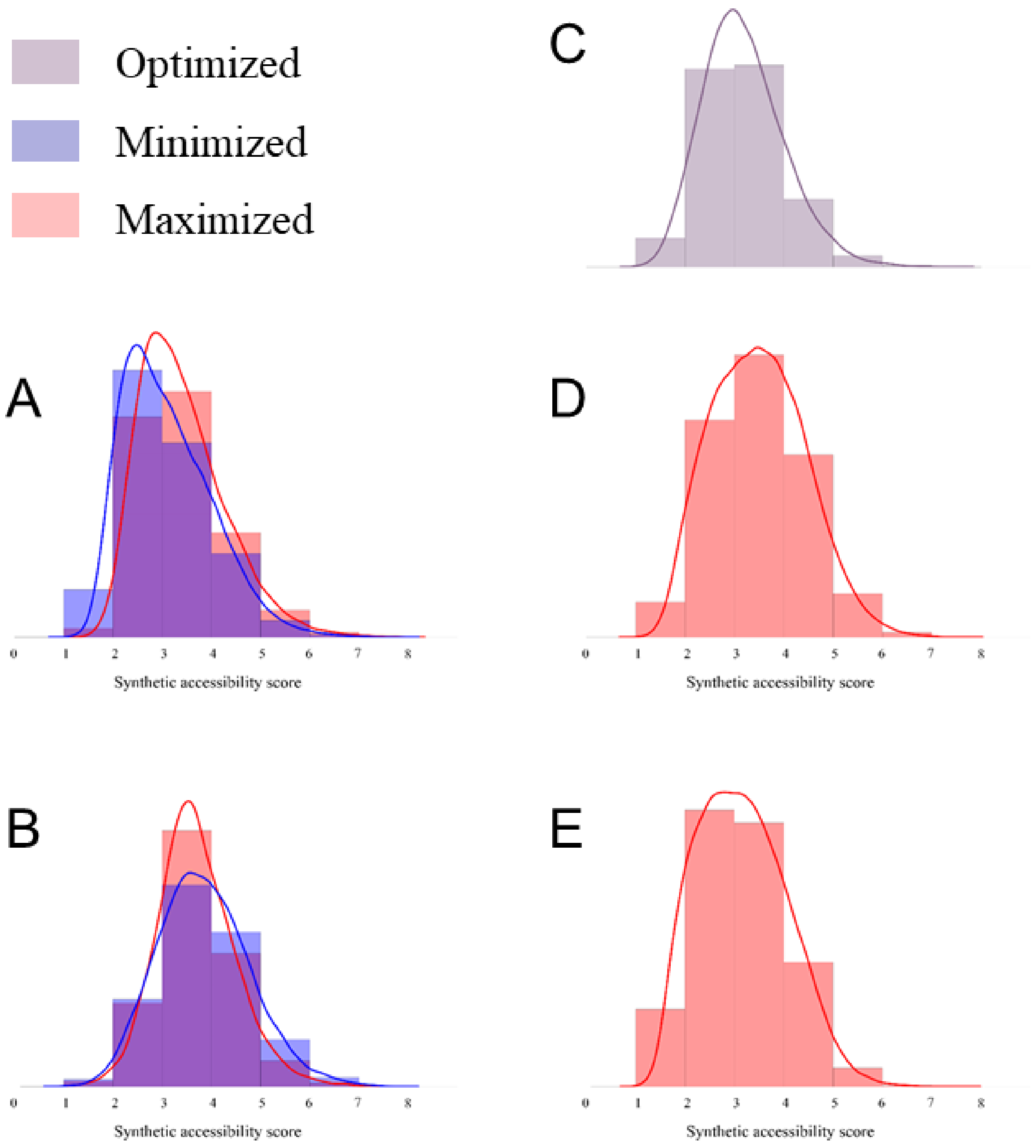

Figure S3. Distributions of Synthetic Accessibility Score for all RL experiments. (A) Melting temperature (B) JAK2 inhibition (C) Partition coefficient (D) Number of benzene rings (E) Number of substituents.

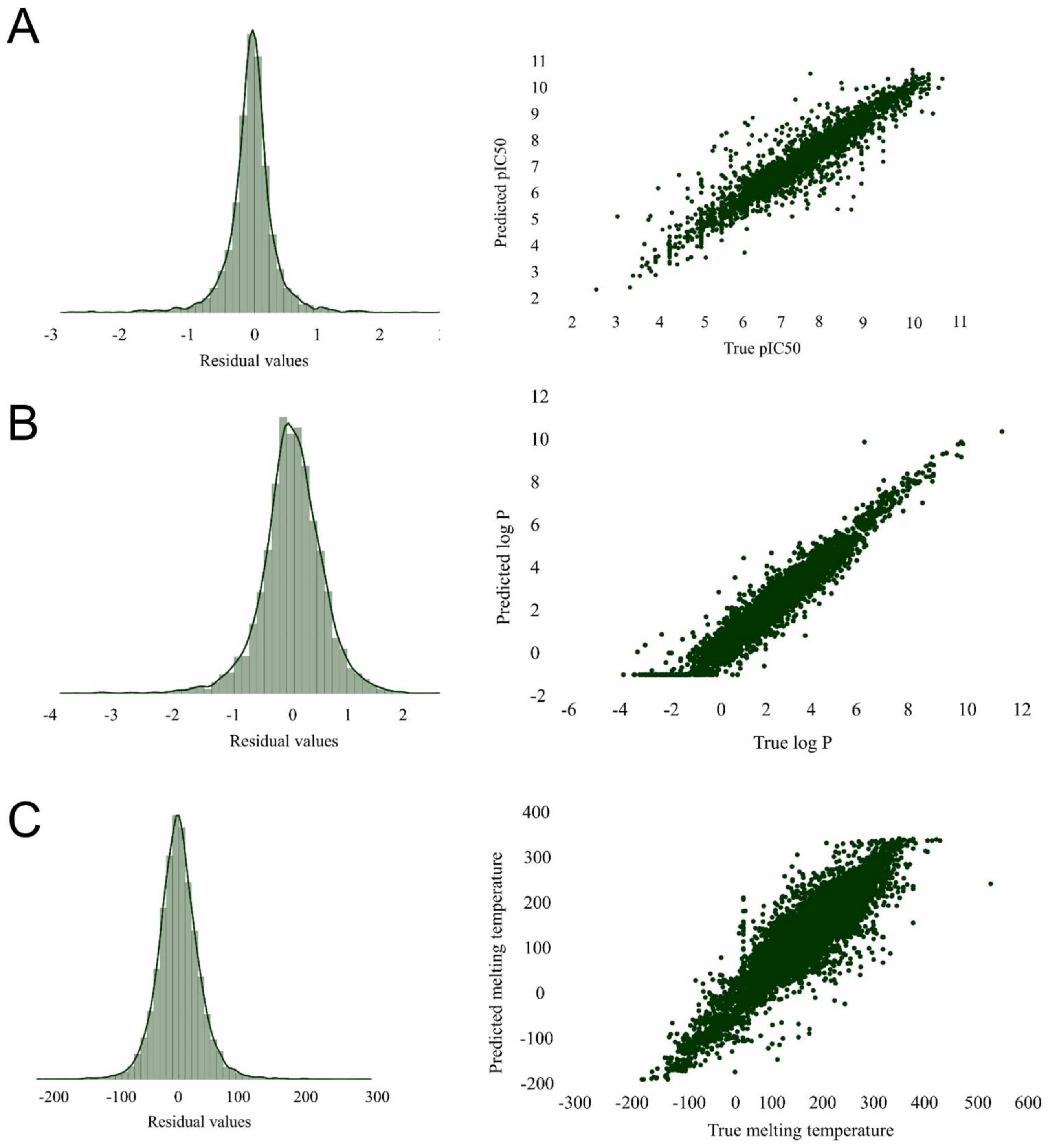

Figure S4. Distribution of residuals and predicted vs. observed plots for predictive models. All the values are calculated on hold out test datasets **(A)** Melting temperature **(B)** logP **(C)** pIC_{50} for JAK2.

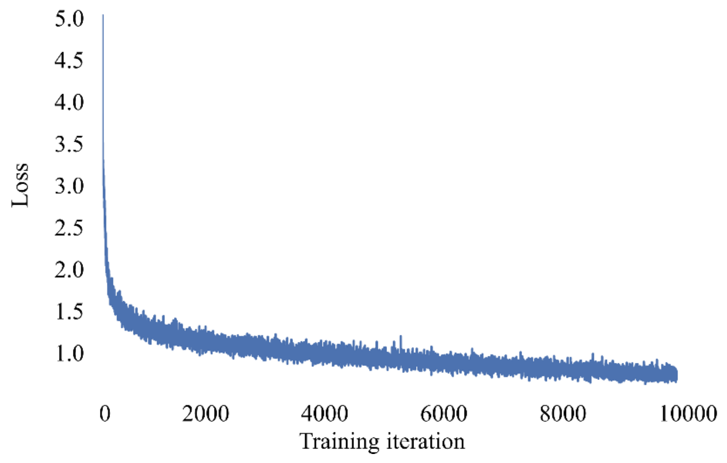

Figure S5. Learning curve for generative model.

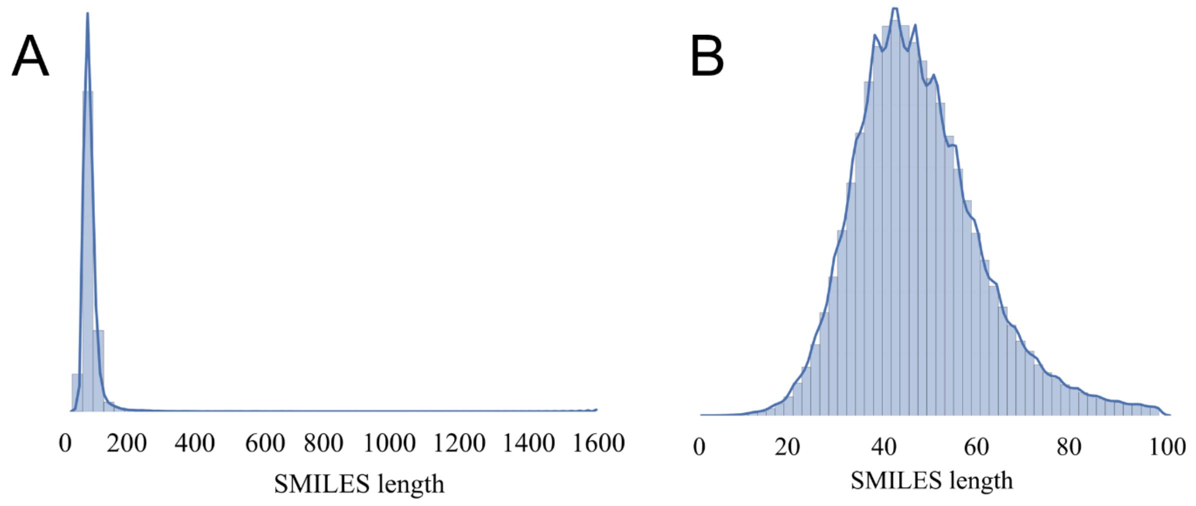

Figure S6. Distributions of SMILES's strings lengths. (A) Initial **(B)** Truncated.