Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики
Вычислительный центр им А. А. Дородницына РАН
Кафедра "Интеллектуальные системы"

03.04.01 Прикладные математика и физика

Порождающие и разделяющие модели для генерации новых лекарств

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

Автор ....

Попова Мария Сергеевна
Кафедра "Интеллектуальные системы"

Научный руководитель ....

Стрижов Вадим Викторович
Доктор физико-математических наук
Научный сотрудник ВЦ РАН

Заведующий кафедрой ....

Рудаков Константин Владимирович
Доктор физико-математических наук
Профессор, академик РАН

Москва
2017

Аннотация

Дизайн лекарств – очень сложный процесс, который требует много времени и средств. От момента начала исследований до того момента, когда новый препарат будет доступен для продажи, может пройти до 25 лет. Процесс дизайна нового лекарства может потерпеть неудачу в любой момент, но чаще всего это происходит на этапах клинических испытаний, когда прошло десятилетие работы и были потрачены миллиарды долларов. Зачастую неудачи случаются, потому что некоторые свойства молекул лекарства оказались неподходящими. Например, молекула оказалась токсичной или биологически неактивной в отношении желаемой мишени. Таким образом, существует проблема выявления молекул-кандидатов с определенными свойствами на самой ранней вычислительной стадии открытия лекарств. В основном существуют две стратегии дизайна лекарств с помощью вычислительных методов. Первая – это виртуальный скрининг, и её цель – поиск перспективных соединений среди миллионов существующих молекул. Вторая стратегия – разработка лекарственного препарата de-novo. Её цель – создать молекулы лекарств с желаемыми свойствами с нуля. В этой работе мы предлагаем новую вычислительную стратегию для de-novo дизайна лекарств, которая основана на глубоком обучении и методах обучения с подкреплением. Эта стратегия позволяет получать химические соединения с требуемыми свойствами. Общий рабочий процесс представлен двумя глубокими нейронными сетями – генеративной и прогностической. Процесс обучения состоит из двух этапов. На первом этапе обе модели обучаются отдельно с помощью алгоритмов обучения с учителем, а на втором этапе модели проходят оптимизируются с помощью обучения с подкреплением. В этом исследовании мы проводим вычислительный эксперимент, который демонстрирует эффективность предлагаемой стратегии. Мы обучаем пять генеративных моделей для генерации химических соединений с оптимизированными физическими, химическими, структурными или биоактивными свойствами. Мы также представляем несколько наборов данных новых соединений.

Оглавление

Список иллюстраций

Список таблиц

Глава 1

Введение

The analysis of recent trends in drug development and approval presents rather bleak picture [1]. The approval of new drugs has been flat over the last two decades. Less than one out of every 10,000 drug candidates will be approved for sales on the market. Only three out of every 20 approved drugs will be profitable enough to cover the costs of their development. Moreover, the process of drug discovery is time and many consuming as an average cost of developing each new drug is \$1-3 billion and it takes approximately 10-15 years. A big number of promising drug candidates fail in later stages of clinical development process – phase II and phase III. At this stage failure is very expensive, as the projects have already incurred high costs. This so-called innovation gap can be attributed to several challenges ranging from drug safety concerns, lack of efficacy to great complexity of diseases and tightened regulations. Thus, pharmaceutical industry is currently challenged to increase the efficiency of drug development. Increasingly scientific advancements are more subject to error and harder to reproduce. Human activities are identified as a principal bottleneck in technological innovations. Which leads to inefficiencies, potential errors, and incomplete explorations of the hypothesis and data analysis space. Artificial intelligence (AI) systems can radically transform the practice of scientific discovery. The combination of artificial intelligence and bis data is sometimes referred to as the fourth industrial revolution [2]. Today as machine learning also enables our computers to teach themselves drive cars or automatically understand speech. AI is revolutionizing meny nedical specialties, for examples radiology and pathology [3, 4]. Application of Deep Learning (DL) see significant improvement in docking scoring [5], learning from small data [6], reaction mechanism [7] and energy prediction [8]. Drug discovery pipeline is notoriously sequential. Hits from a high throughput screen (HTS) are slowly progressed toward promising lead compounds. Next ADMET and selectivity profile is optimized with a challenge to maintain high potency and efficacy. High failure rates in late-stage clinical trials could be potentially avoided if the essential information were provided earlier or if the provided data could give some understanding whether a drug-candidate will actually have all the expected properties in clinical practice. The crucial step is to formulate of a well-motivated hypothesis for the problem compound generation (de novo design) or compound picking from a library using the available SaR data. Commonly, when a team of scientists from multiple disciplines generates the new hypothesis, it usually relies on the medicinal chemistry intuition and expertise of the team members. Therefore, any design hypothesis is easily biased towards preferred chemistry [9] or model interpretation [10]. Idea of automated drug design is not new [11, 12]. It has been used in drug discovery projects since 2000s by constructing novel molecules with desired properties from scratch. This idea has already become an active research field. In an attempt to design new compounds, both a medicinal chemist and algorithm is confronted with a virtually infinite chemical space. Today range of potential drug-like molecules is estimated to be between 10^{30} and 10^{60} [13, 14]. Unfortunately, high-throughput screening (HTS) technology is not applicable to perform a search in such a large space [15]. Instead of the systematic construction and evaluation of each individual compound, *de novo* design process exploits principles of local optimization, which means, that the process does not necessarily converges to the optimal solution, but leads to a local or 'practical' optimum by stochastic sampling, or restricts the search to a smaller subspace of the whole chemical space which can be screened exhaustively [11, 16, 17, 18]. However, recently a method for exploring chemical space based on continuous encodings of molecules was proposed [19]. It allows directed gradient-based search through chemical space. Here we propose a novel method based on deep reinforcement learning (RL) for generating chemical compounds with desired physical, chemical or bio activity properties. Reinforcement learning (RL) is a subset of artificial intelligence which is used to solve dynamic decision problems. RL involves analyzing possible actions, estimating the statistical relationship between the actions and their possible outcomes and then determining a treatment regime that attempts to find the most desirable outcome based on the analysis The integration of reinforcement learning and neural networks dated back to 1990s [20]. However, with recent achievements of DL, benefiting from big data, new powerful algorithmic approaches are emerging. We are currently witnessing the renaissance of reinforcement learning [21], especially, the combination of reinforcement learning and deep neural networks, i.e., deep reinforcement learning (Deep RL). Most recently RL has led to superhuman performance in game Go [22], considered practically intractable due to the theoretical complexity of over 10^{140} [23]. Therefore, we see an example of attacking a problem of the difficulty comparable to a chemical space exploration without brute-force computing every possible solution.

Глава 2

Методы

In this work we propose a novel RL based de novo design method for generating chemical compounds with desired physical, chemical or bio activity properties. The general workflow (see Figure 2-1) is represented by two deep neural networks (generative G and predictive D). 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 models are trained jointly with reinforcement learning approach optimizing target properties. In this system generative models plays the role of agent. Predictive model plays the role of critic, which estimates the agent's behavior by assigning a numerical reward to every generated molecule. The reward is function of the numerical property predicted by the predictive model. The generative model is trained to maximize the expected reward.

The first model in our methodology is a generative recurrent neural network [24, 25, 26], which outputs molecules in the simplified molecular-input line-entry system (SMILES) notation [27]. We propose using a special type of RNN, which is stack-augmented recurrent

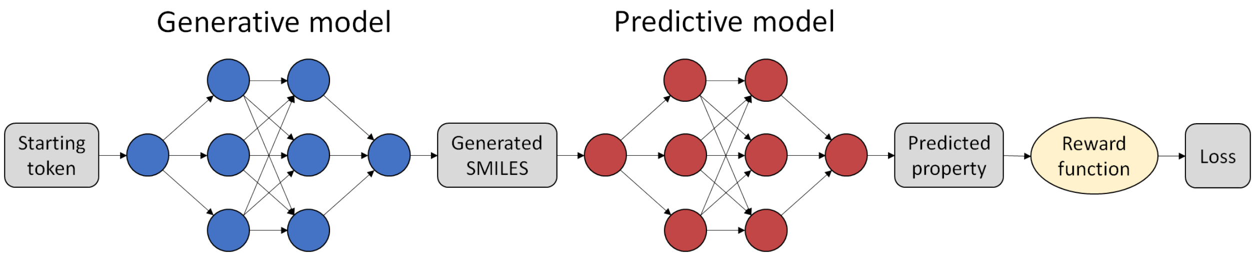
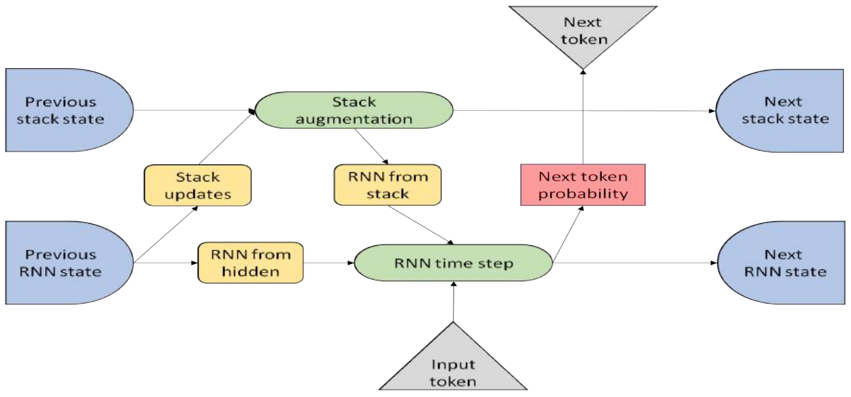
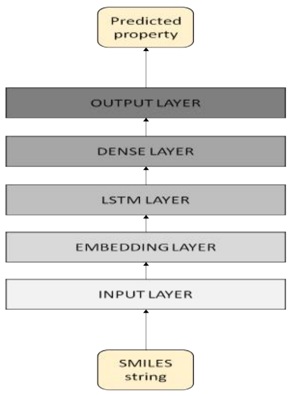


Рис. 2-1: General pipeline of Reinforcement learning system for novel compounds generation



(a) Scheme of stack augmented RNN time step



(b) Scheme of predictive network

Рис. 2-2: Generative and predictive networks

neural network [28] (see Figure (2-2a)). Neural networks with stack memory are capable of solving problems of sequence prediction which are unsolvable with regular neural networks. This is possible because stack memory provides such neural networks the capacity to count and memorize the sequences. One of the examples of sequences which can not be properly modeled by regular recurrent network is words from Dyck language, which is a language of balanced strings of brackets [29]. Another weakness of regular recurrent neural networks is their inability to capture long term dependencies which leads to difficulties in generalizing to longer sequences [30]. All of these properties are required to learn language of 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 sequence with several bracket types. Therefore, Stack RNNs are proper choice for modeling such sequence dependencies.

The second model in our methodology is a predictive model for estimating physical, chemical or bio activity properties of molecules. This property prediction model is a deep neural network, which consists of embedding layer [31], LSTM layer [32] and two dense layers. This network is designed to designed to calculate user-specified property (activity) of the molecule taking tokenized SMILES string as an input data vector. The advantage of such approach is that no numerical descriptors are needed as it learns directly from SMILES notation.

At first stage, we pretrain a generative model on a dataset of approximately 1.5M drug-like compounds, so that the model is capable of producing chemically feasible molecules, but without property optimization. At second stage we combine both generative and predictive model into one reinforcement learning system. In this system generative plays the role of agent, whose action space is represented by the SMILES notation alphabet and state space is represented by all possible strings in this alphabet. Predictive model plays the role of critic, which estimates the agent's behaviour by assigning a numerical reward to every generated molecule. The reward is function of numerical property predicted by predictive model. At this stage the generative model is trained to maximize the expected reward. The whole pipelene is illustrated on Figure (2-1).

2.1 Постановка задачи

This section provide a formal problem statement for the task of generating SMILES strings for chemical compounds with desired properties.

Given a set of objects $\mathbb{S} = \{\mathbf{s}_1, \dots, \mathbf{s}_n\}$, where each object $\mathbf{s}_i$ is a string of characters from alphabet $\mathbf{A} = \{a_1, a_2, \dots, m\}$. Without loss of generality, the length of each sequence from set $\mathbb{S}$ is assumed to be fixed and equal to L. Let's assume that there exists a probability distribution p over the set $\mathbf{A}^L$, which is a set of all possible sequences of length L from alphabet $\mathbf{A}$, and that the set $\mathbb{S}$ is sampled from the distribution p. The problem then can be formulated as follows: construct a model for sampling new objects from distribution p.

Let's assume that the stated problem can be solved by a parametric generator model $G(\boldsymbol{\theta}; \mathbf{w})$, which will be described further in details, where $\boldsymbol{\theta} \in \mathbb{R}^m$ is a vector of parameters and $\mathbf{w} \in \boldsymbol{\Theta}$ is a vector of hyper-parameters. The vector w of hyper-parameters is assumed to be fixed. Further we will call G as *generative* model and omit vector $\mathbf{w}$ of hyper-parameters in its notation. Generative model $G(\boldsymbol{\theta})$ is probabilistic and can be applied to the task of generating new objects:

$$\hat{\mathbb{S}} = \mathbf{f}(\mathbf{w}; \boldsymbol{\theta}),$$

where $\hat{\mathbb{S}} = \{\hat{\mathbf{s}}_1, \dots, \hat{\mathbf{s}}_m\}$ is a set of generated by the generator model objects such that $\hat{\mathbf{s}}_j \notin \mathbb{S}\ \forall j = 1, \dots, m$. The optimal vector of parameters $\hat{\boldsymbol{\theta}}$ of the generator model $\mathbf{f}$ can be found by optimizing two functions:

$$\begin{cases} L(\mathbb{S}, G) \to \min\limits_{\boldsymbol{\theta}}, \\ Q(\hat{\mathbb{S}}) \to \max\limits_{\boldsymbol{\theta}} \end{cases} \to \hat{\boldsymbol{\theta}},$$

where

$$L(\mathbb{S}, G) = \sum_{i=1}^{n} l(\mathbf{s}_i, G)$$

is a sum of loss functions on training set and

$$Q(\hat{\mathbb{S}}) = \sum_{i=1}^{m} q(\hat{\mathbf{s}}_i)$$

is a sum of quality functions on generated set. Both functions will be particularly defined further.

2.2 Порождающая модель как генеративная рекуррентная нейронная сеть со стековой памятью

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, for example, characters. The goal is to build a model, which is able to predict the next token taking all the previous ones. 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:

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

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 ans 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:

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

where D is $1 \times m$ matrix and $a_t = [a_t[\text{PUSH}], a_t[\text{POP}], a_t[\text{NO-OP}]]$ is a vector of stack control variables, which define the next operation to be performed. If $a_t[\text{POP}]$ is equal to 1, then the value below is used to replace the top element of the stack. If $a_t[\text{PUSH}]$ is equal to 1, then a new value will be added to the top and all the rest values will be moved down. If $a_t[\text{NO-OP}]$ equals 1 then stack keeps the same value on top. Similar rule is applied to the elements of the stack at a depth $i > 0$:

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

Now the hidden layer h_t is updated as:

$$h_t = \sigma(Ux_t + Rh_{t-1} + Ps^k_{t-1}),$$

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

2.3 Разделяющая модель для оценки свойств

This section describes predictive model D that is used to estimate quality Q of generated molecules. In this model we also consider sequences of discrete tokens, for examples, characters. The goal of predictive model is to estimate a given property taking this sequential data as an input. As it was mentioned above, predictive model D is a deep neural network, that consists of input layer, embedding layer [31], long-short term memory layer [32] and 2 feed-forward layers. The architecture of predictive model is illustrated on Figure 2-2b.

Input layer takes a sequence of discrete tokens, embedding layer processes this sequence into a continuous vector of representations. Further goes LSTM layer with $tanh$ non-linearity function, that transform vector from embedding layer into a feature vector. The first feed forward layer performs nonlinear $tanh$ transformation of the feature vector and the next feed-forward layer with a $relu$ activation function and one unit predicts a desired property.

2.4 Поставнока задачи молекулярного дизайна как задачи обучения с подкреплением

This section describes how the stated problem of new SMILES generation can be formulated in terms of Reinforcement Learning approach. The idea is to combine both generative G and predictive model D into one reinforcement learning system. The set of actions A is defined as an alphabet of SMILES notation. The set of states S is defined as all possible strings in the alphabet with lengths from 0 to some T. The state s_0 with length 0 is unique and considered to be an initial state. The state s_T of length T is called terminal state and it causes episode to end. The subset of terminal states $S^* = \{s_T \in S\}$ of S which contains all the states s_T with length T is called the terminal states set. Reward $r(s_T)$ is calculated in the end of an episode, when 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 previous state s_{t-1} as an input and estimates probability distribution $p(a_t|s_{t-1})$ 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 D:

$$r(s_T) = f(D(s_T)),$$

where f is chosen expertly depending on the task. Some examples of the functions f are provided further 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:

$$J(\theta) = \mathbb{E}[r(s_T)|s_0, \theta] = \sum_{s_T \in S^*} G(s_T) r(s_T) \to \max.$$

This sum iterates over the set S^* of terminal states. In our case this set is exponential and the sum can not be exactly computed. The trick is to approximate this sum as a mathematical expectation by sampling terminal sequences from the model distribution:

$$J(\theta) = \mathbb{E}[r(s_T)|s_0, \theta] = \mathbb{E}_{a_1 \sim p_\theta(a_1|s_0)} \mathbb{E}_{a_2 \sim p_\theta(a_2|s_1)} \dots \mathbb{E}_{a_T \sim p_\theta(a_T|s_{T-1})} r(s_T).$$

So, the procedure for $J(\theta)$ estimation is following: sequentially sample a_t from the model G for t from 0 to T. The unbiased estimation for $J(\theta)$ is the sum of all rewards in every time step which in our case equals to the reward for the terminal state as we assume that intermediate rewards are equal to 0. As this quantity needed to me maximizes, we need to compute its gradient. This can be done with a REINFORCE algorithm [33] which uses approximation of mathematical expectation as a sum, which we provided above, and the following trick:

$$\partial_\theta f(\theta) = f(\theta) \frac{\partial_\theta f(\theta)}{\partial \theta} = f(\theta) \partial_\theta [\log f(\theta)].$$

So, the gradient of $J(\theta)$ can be written down as:

$$\partial_\theta J(\theta) = \sum_{s_T \in S^*} [\partial_\theta p_\theta(s_T)] r(s_T) =$$

$$= \sum_{s_T \in S^*} p_\theta(s_T) [\partial_\theta \log p_\theta(s_T)] r(s_T) = \sum_{s_T \in S^*} p_\theta(s_T) \left[\sum_{t=1^T} \partial_\theta \log p_\theta(a_t|s_{t-1}) \right] r(s_T) =$$

$$= \mathbb{E}_{a_1 \sim p_\theta(a_1|s_0)} \mathbb{E}_{a_2 \sim p_\theta(a_2|s_1)} \dots \mathbb{E}_{a_T \sim p_\theta(a_T|s_{T-1})} \left[\sum_{t=1^T} \partial_\theta \log p_\theta(a_t|s_{t-1}) \right] r(s_T),$$

which gives as an algorithm for $\partial_\theta J(\theta)$ estimation.

Глава 3

Вычислительный эксперимент

3.1 Описание данных

3.1.1 Данные для порождающей модели

For training generative model G we took ChEMBL database of drug-like compounds [34], which consists of approximately 1.5 million of SMILES strings.

We preprocessed the data by selecting from initial training dataset just those molecules, which SMILES notation length is less than 100 characters. The length of 100 is chosen because more than 97% of SMILES in training dataset are 100 characters or less (see Figure 3-1).

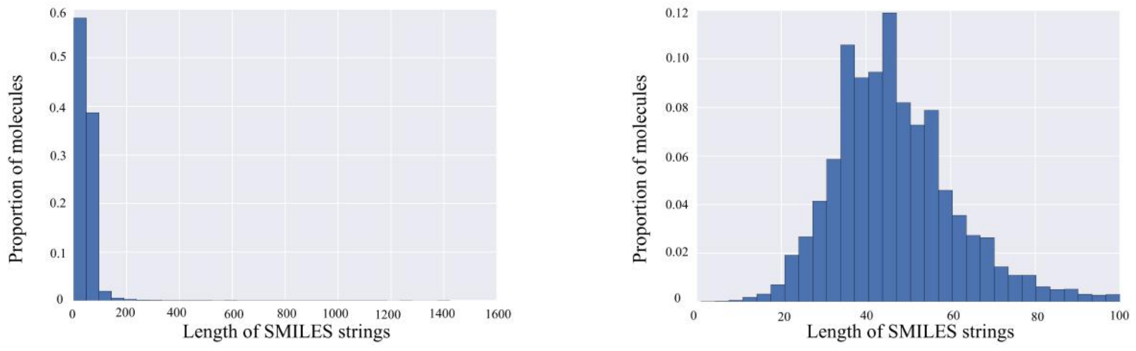


Рис. 3-1: Initial (left) and truncated (right) distribution of SMILES's lengths

3.1.2 Данные для разделяющей модели

We have dataset for three properties – melting temperature, partition coefficient log P and pIC50 of JAK2 kinase. These datasets include 47000, 15000 and 15000 compounds respectively. Every compound is labeled with a corresponding property.

3.2 Процедура обучения

We trained a stack-augmented RNN which was described in section 2.1. as a generative model. This network has 1500 units in recurrent GRU layer [35] and 512 units in stack augmentation layer. As a training dataset we took ChEMBL database of drug-like compounds [34], which includes approximately 1.5 million of SMILES strings. The model was trained on GPU for 10000 epochs. The learning curve is illustrated in Figure 3-2.

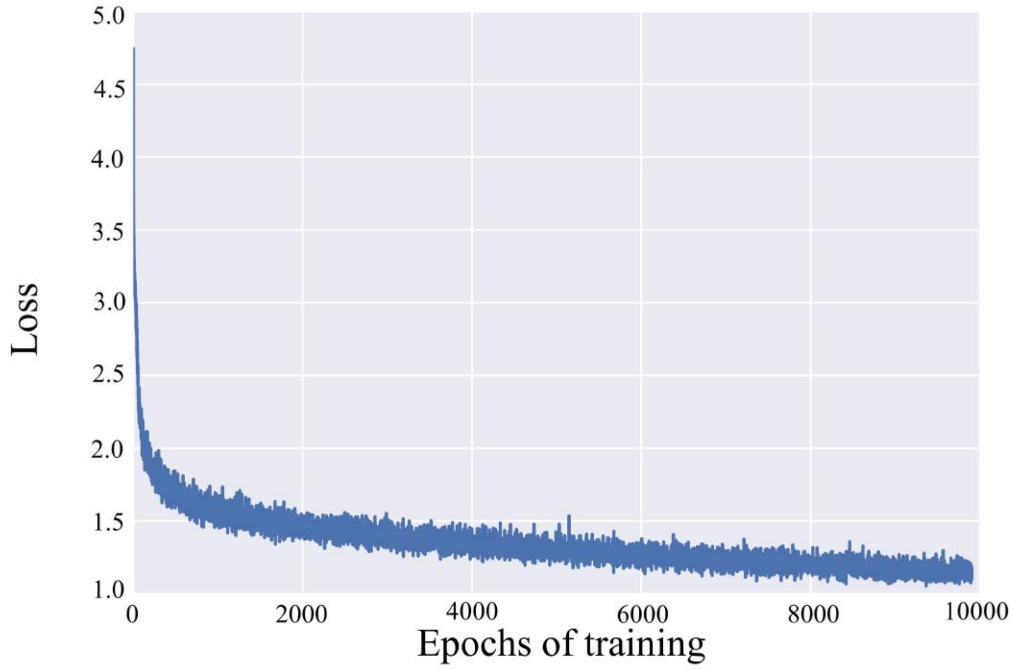


Рис. 3-2: Learning curve of generative model

As an object for training our generative model receives a sequence of 100 characters, which can include several SMILES strings, separated by spaces. The last SMILES in a training sequences is truncated, so that the length of the whole sequence doesn't exceed 100 characters.

3.3 Порождающая модель

The generative model has two modes of processing sequences – training and generating. In training mode at each time step the generative network takes a current prefix of training object and predicts the probability distribution of 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. In generating mode at each time step the generative network takes a prefix of already generated sequence and then, similar to training mode, predicts probability distribution of next character and samples it from this predicted distribution. In generating mode we do not update model parameters.

3.3.1 Генерация неоптимизированных молекул

To demonstrate the versatility of the baseline (unbiased) Stack RNN we generated a dataset of over one million virtually synthesized compounds. Random examples of the generated compounds are illustrated in Figure 3-3. Over 91% of generated structures, were valid chemically-sensible molecules. The validity check was performed by the structure checker from ChemAxon [36]. When compared with ChEMBL [34], model produced just about 1% of structures from the training dataset. Additional comparison with ZINC15 database [37] of 320M synthetically accessible drug-like molecules show match of about 4% structures. Overall, this analysis suggests that generative Stack RNN model did not simply memorized training SMILEs sequences but is capable to generate extremely diverse but realistic molecules.

3.4 Разделяющая модель

We trained three predictive models for three different properties – melting temperature, log P and pIC50 for JAK2 kinase. Each model consists of embedding layer, which transforms sequence of discrete tokens into a vector of 100 continuous numbers, LSTM layer with 100

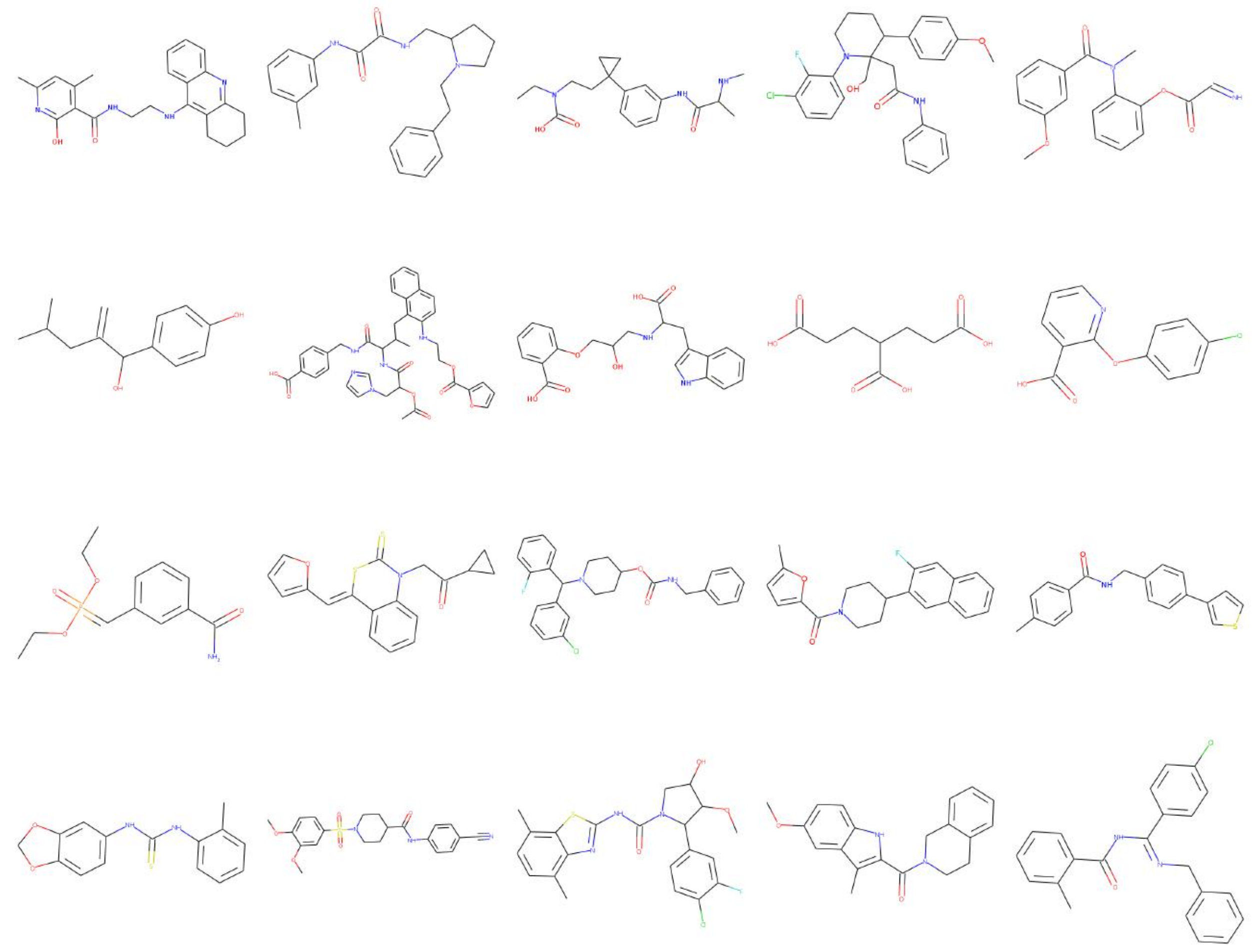


Рис. 3-3: Examples of molecules produced by generative model

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. As it was mentioned above, the training datasets for melting temperature, log P and pIC50 for JAK2 kinase consist of 47000, 15000 and 15000 compounds respectively. These datasets were divided into training and validation sets in a ratio of 3 : 1. The results and accuracy of the model are shown in Figure 3-4.

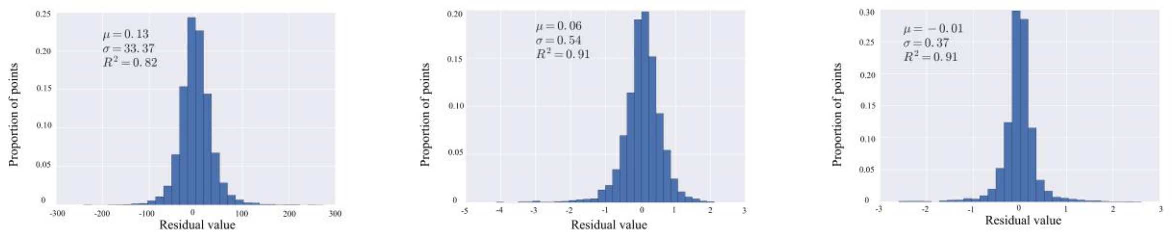


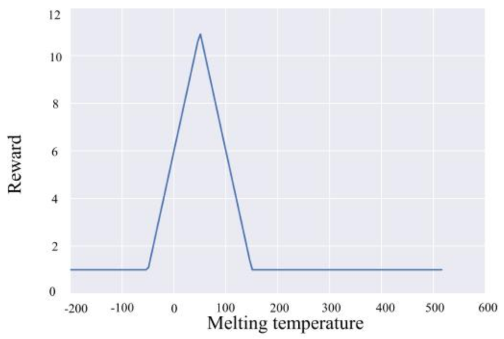
Рис. 3-4: Distribution of residuals for predictive models

3.5 Система обучения с подкреплением

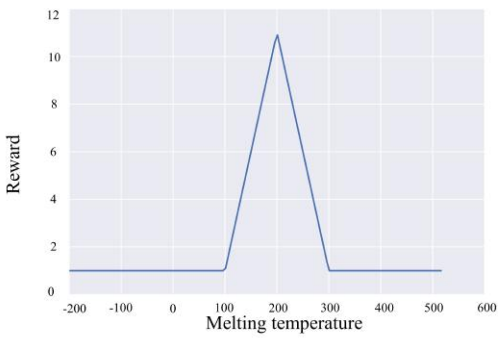
To explore the utility of the RL algorithm in a drug design setting we have set up a multiple case studies that optimize three types of rewards: a) physical property, b) biological activity and c) chemical substructure. For physical properties we selected melting temperature (Tm) and octanol-water partition coefficient (logP). Inhibition potency in form of IC50 to JAK2 kinase was used as biological activity. Finally, number of benzene rings and number of substituents (like –OH, -NH2, -CH3 –CN, etc.) was used as a structural reward. Figures 3-6, 3-9, 3-13, 3-16a, and 3-16b show distribution of predicted properties of interest before and after experiments. In both cases, we sampled 10000 molecules by the default and optimized generative models and calculated their properties with a corresponding predictive model. Values of the substructure features were calculated directly from the 2D structure. Table 4.1 summarizes analysis of generated molecules and descriptive statistics.

3.5.1 Оптимизация температуры плавления

In this experiment we set two goals to minimize and maximize the target property. Upon minimization the mean of the distribution was shifted by 44^oC. Optimized generator virtually synthesized simple hydrocarbons like butane, and poly-halogenated compounds CF_2Cl_2 and $C_6H_4F_2$. CF_4 molecule has a lowest $T_m = -184^oC$ in the produced dataset. This property minimization strategy is extremely effective, it allowed to discover molecules in the regions of chemical space far beyond available in the training examples. In the maximization regime mean of the melting temperature is increased by 20^oC to 200^oC. Generator synthesized substantially more complex molecules with abundance of Sulphur heterocycles, phosphates as well as conjugated double bonds. The reward functions in both cases are defined as piecewise linear function from melting temperature (see Figure 3-5). Figure 3-6 demonstrates the results of optimization and Figures 3-7 and 3-8 provide examples of generated molecules both for minimization and maximization regimes.



(a) Reward function for melting temperature minimization



(b) Reward function for melting temperature maximization

Рис. 3-5: Reward function for melting temperature optimization

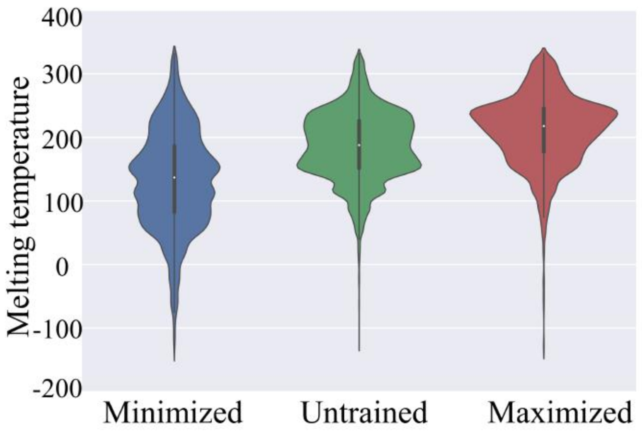


Рис. 3-6: Distribution of minimized, untrained and maximized melting temperature

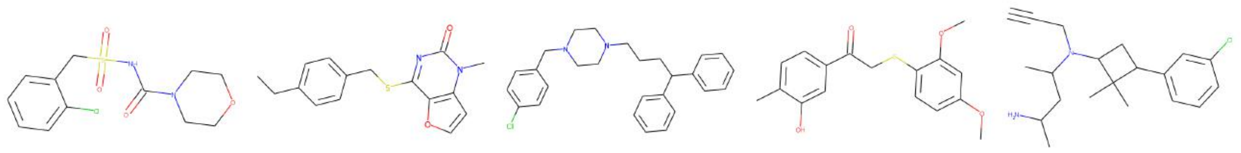


Рис. 3-7: Examples of molecules with minimized melting temperature

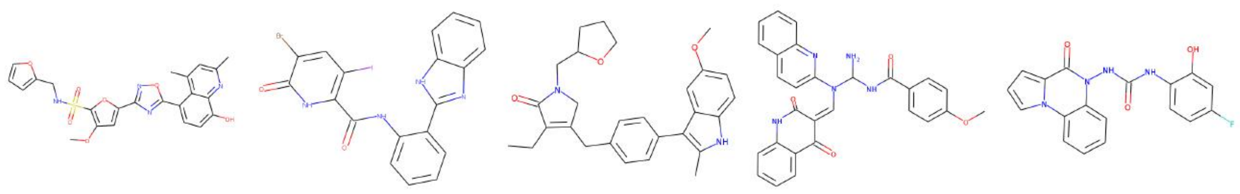


Рис. 3-8: Examples of molecules with maximized melting temperature

3.6 Оптимизация липофильности

In the second experiment we set the goal to optimize the log P property values of generated molecules. To better mimic requirements of drug-likeliness instead of property minimization we imposed to the range. The reward function in this case was defined as a piecewise linear function of log P with a constant region from 1.0 to 4.0 (see Figure 3-10). In other words, we set the goal to uniformly synthesize molecules according to a typical Lipinski's constraints. After training 88% of generated molecules were within logP from 0 to 5. The results of optimization are demonstrated in Figure 3-9. Figure 3-11 show some examples of generated molecules.

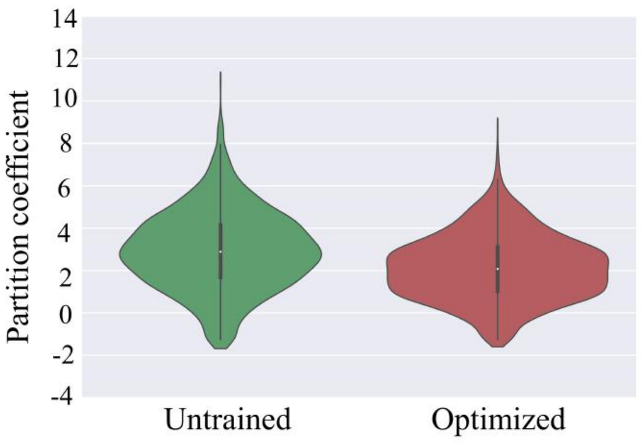


Рис. 3-9: Distribution of untrained and optimized log P

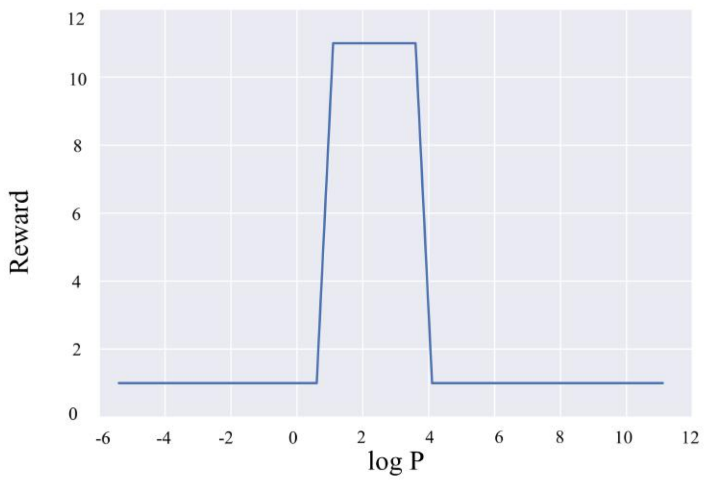


Рис. 3-10: Reward function for log P optimization

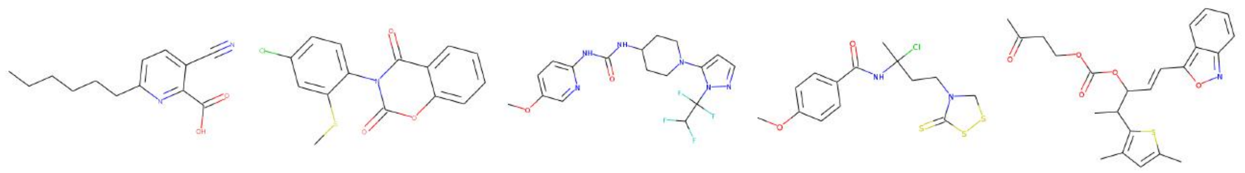
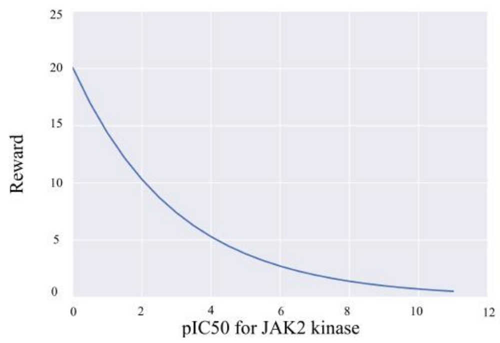


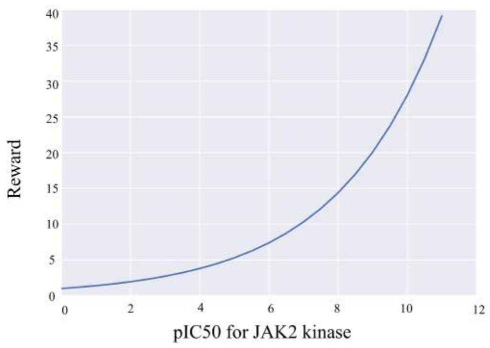
Рис. 3-11: Examples of molecules with optimized partition coefficient

3.7 Оптимизация биологической активности

In the third experiment, perhaps most relevant to the practical drug discovery application we directly minimized and maximized pIC50 values for JAK2 kinase. The reward function both in cases was defined as exponential functions of pIC50 (see Figure 3-12). The results of optimization are demonstrated in Figure 3-13. With minimization, the mean of predicted pIC50 distribution was shifted by about one unit. However, distribution is heavily tailed, and 24% of molecules are predicted to have practically no activity (pIC50 $\leq$ 4). In the maximization strategy, properties of generated molecules were more tightly distributed bet. In both cases models virtually synthesized known and novel compounds based on one scaffold as well as suggested new scaffolds. Overall, system retrospectively discovered multiple commercially available compounds deposited in ZINC database. Figures 3-14 and 3-15 show some examples of generated molecules both for JAK2 activity minimization and maximization.



(a) Reward function for pIC50 of JAK2 kinase minimization



(b) Reward function for pIC50 of JAK2 kinase maximization

Рис. 3-12: Reward function for pIC50 of JAK2 kinase optimization

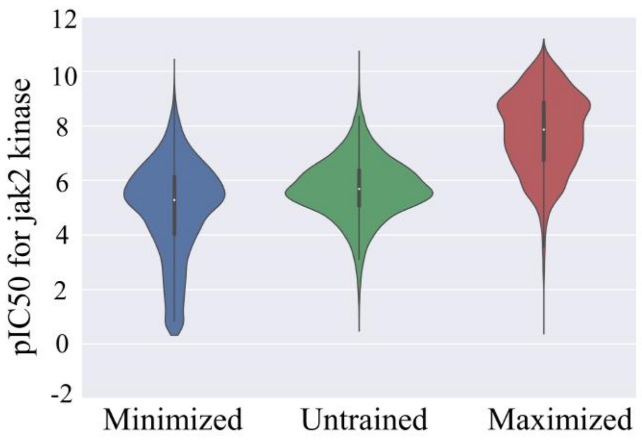


Рис. 3-13: Distribution of minimized, untrained and maximized pIC50 of JAK2 kinase

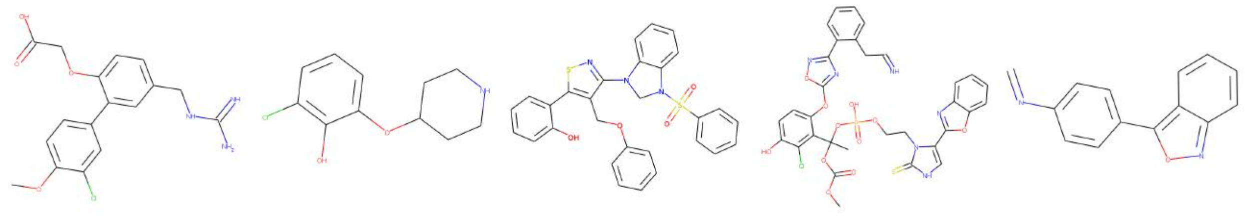


Рис. 3-14: Examples of molecules with minimized pIC50 for jak2 kinase

3.8 Оптимизация структурных свойств

Finally, we also performed two simple experiments mimicking biasing chemical libraries to a user-defined substructure without predicting any property. We defined the reward function as the exponent of a) number of monosubstitured benzene rings (-Ph) (see Figure 3-17a)

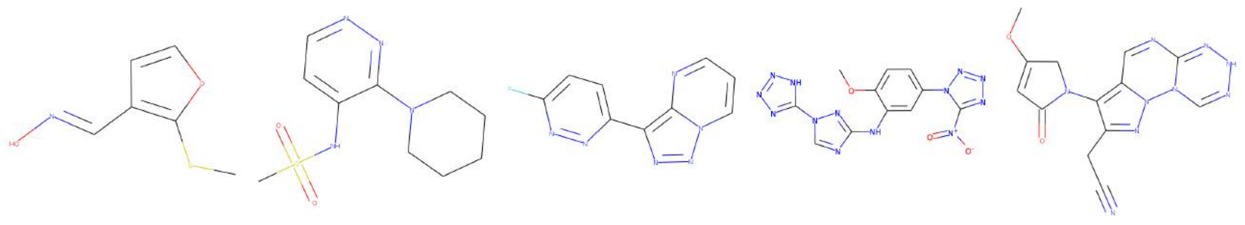
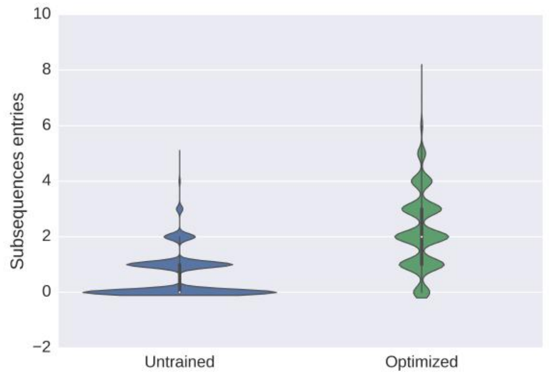
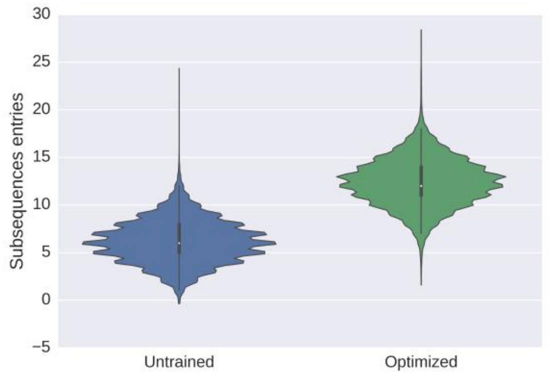


Рис. 3-15: Examples of molecules with maximized pIC50 for jak2 kinase

and b) total number of small groups substituents (see Figure 3-17b). Among all case studies described, structure bias was easiest to optimize. Figures 3-16a and 3-16b illustrate results of optimization and Figures 3-18 and 3-19 show some examples of generated molecules.

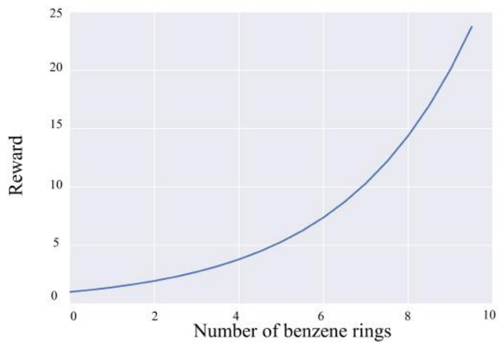


(a) Distribution of untrained and maximized number of benzene rings

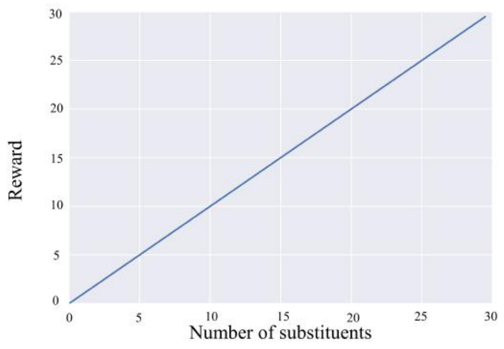


(b) Distribution of untrained and maximized number of substituents

Рис. 3-16: Distributions of structure bias optimization



(a) Reward function for benzene rings number maximization



(b) Reward function for substituents number maximization

Рис. 3-17: Reward function for structure bias optimization

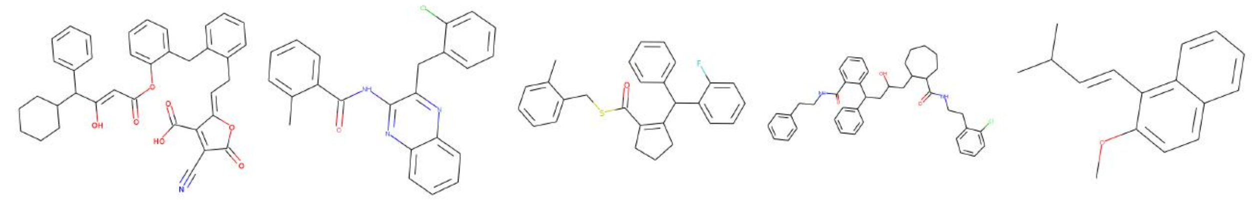


Рис. 3-18: Examples of molecules with maximized number of 6 carbon ring

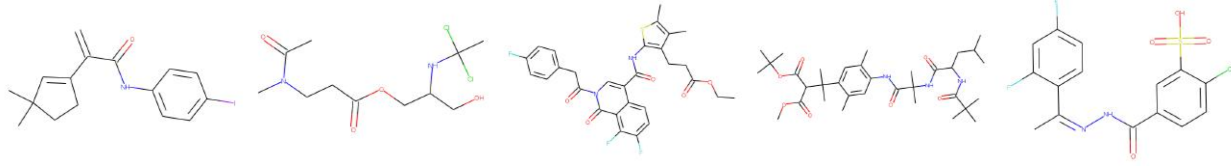


Рис. 3-19: Examples of molecules with maximized number of hydrogen substituents

Глава 4

Обсуждение и выводы

4.1 Результаты и анализ модели

Table 4.1 summarizes results of all the experiments conducted in the thesis. This table shows decrease in a proportion of valid molecules after optimization. We explain this phenomenon by the weaknesses of predictive model D. I.e. generative model G tends to find some local optima of reward function, that correspond to invalid molecules, but predictive model D assigns these molecules high rewards. Our explanation is supported by the results of structure bias optimization experiments, as in these experiments we didn't use any predictive models and decrease in proportion of valid molecules wasn't so significant. We also notice, that among the experiments which include predictive models, experiment with log P optimization and bioactivity of JAK2 kinase show higher proportion of valid molecules and, at the same time, corresponding predictive models have higher quality $R^2 = 0.91$.

4.2 Интерпретация параметров рекуррентной нейронной сети

In this section we demonstrate how recurrent neural network can memorize and process some properties of the SMILES string that it is currently processing. We looked inside the neurons gate activations of the neural network while it processes the input data (see

Таблица 4.1: Comparison of statistics for optimized, untrained and training molecules datasets

Property		Proportion of valid molecules	Mean molar mass	Mean property value through dataset	Proportion of matches with ZINC database	Proportion of matches with ChEMBL database
Melting temperature	untrained	91%	435.4	181.30	4.7%	1.5%
	minimized	31%	279.6	137.17	4.6%	1.6%
	maximized	53%	413.2	200.715	2.4%	0.9%
pIC50 of jak2 kinase	untrained	91%	435.4	5.70	4.7%	1.5%
	minimized	60%	481.8	4.89	2.5%	1.0%
	maximized	45%	275.4	7.85	4.5%	1.8%
log P	untrained	91%	435.4	3.63	4.7%	1.5%
	optimized	70%	369.7	2.58	5.8%	1.8%
Number of benzene rings	untrained	91%	435.4	0.59	4.7%	1.5%
	optimized	83%	496.0	2.41	5.5%	1.6%
Number of substituents	untrained	91%	435.4	3.8	4.7%	1.5%
	optimized	80%	471.7	7.93	3.1%	0.7%

Figure 4-1). In this figure each line corresponds to activations of one neuron at different time steps of processing SMILES string. Each letter is coloured according to the value of activation in cool-warm colormap from dark blue to dark red – from -1 to 1. We discovered that that our RNN has several interpretable cells, that can be divided into two groups – chemically sensible, that captures chemical groups, such as aromatic moiety, carbonyl group or heterocyclic nitrogen, and syntactic, that, for example, captures brackets or decides when the molecule ends. We also discovered, that some neurons have opposite ones, which get deactivated when processing the same group.

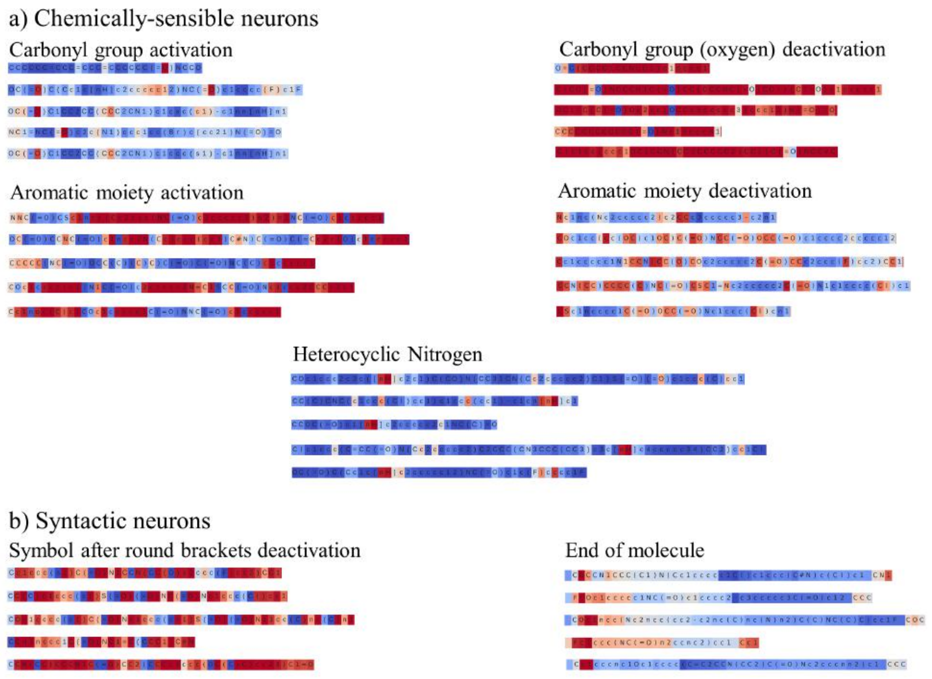

a) Chemically-sensible neurons
Carbonyl group activation
Carbonyl group (oxygen) deactivation
Aromatic moiety activation
Aromatic moiety deactivation
Heterocyclic Nitrogen
b) Syntactic neurons
Symbol after round brackets deactivation
End of molecule

Рис. 4-1: Examples of RNN neurons activation values

4.3 Визуализация результатов

In this section we visualize generated molecules in chemical space using t-Distributed Stochastic Neighbor Embedding (t-SNE) technique for dimensionality reduction [38]. We generated datasets for melting temperature, bio activity of JAK2 kinase and log P with corresponding optimized generative models G, then for every molecule we calculated a vector of representation as an output from the feed-forward layer with $relu$ activation function in the predictive model D for the corresponding property and calculated its 2D projection using t-SNE. Obtained projections are illustrated in Figures 4-2, 4-3, 4-4. In this figure every point corresponds to a molecule and is colored according to its property value in a cool-warm colormap, where dark blue color corresponds to low values and dark red – to high values. For bioactivity of JAK2 kinase and log P t-SNE diagrams have well defined clusters, while for melting temperature there are no such clusters. This observation can be explained by the fact, that melting temperature depends not only on a structure of one separate molecule, but also on intermolecular forces and how the molecules are packed in a substance.

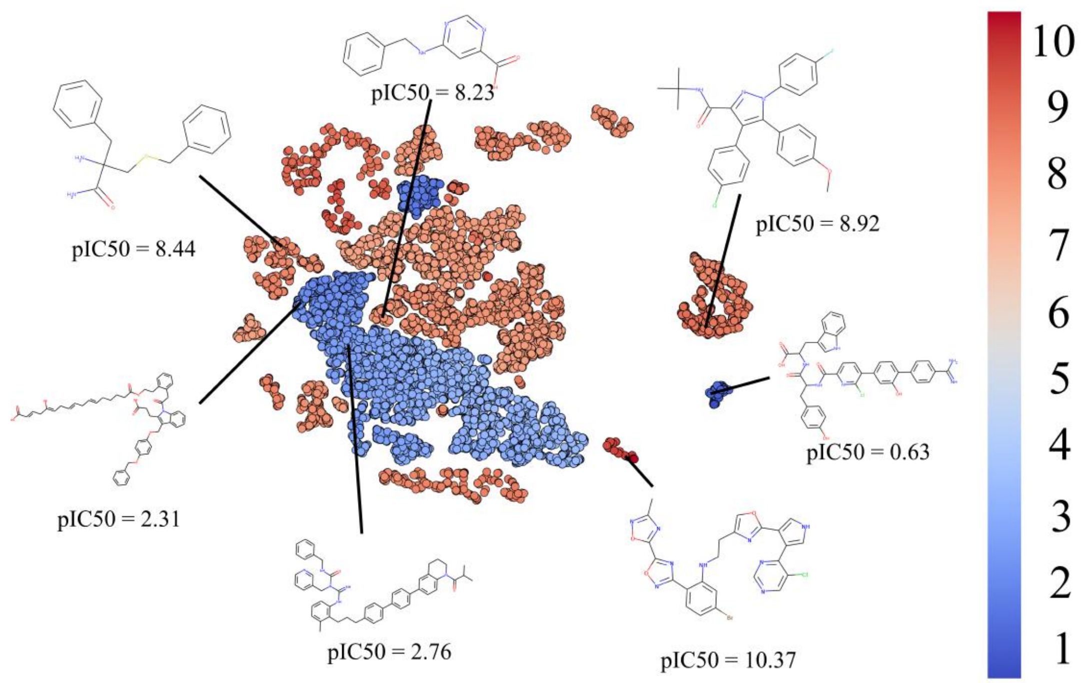


Рис. 4-2: t-SNE diagram for bioactivity of JAK2 kinase

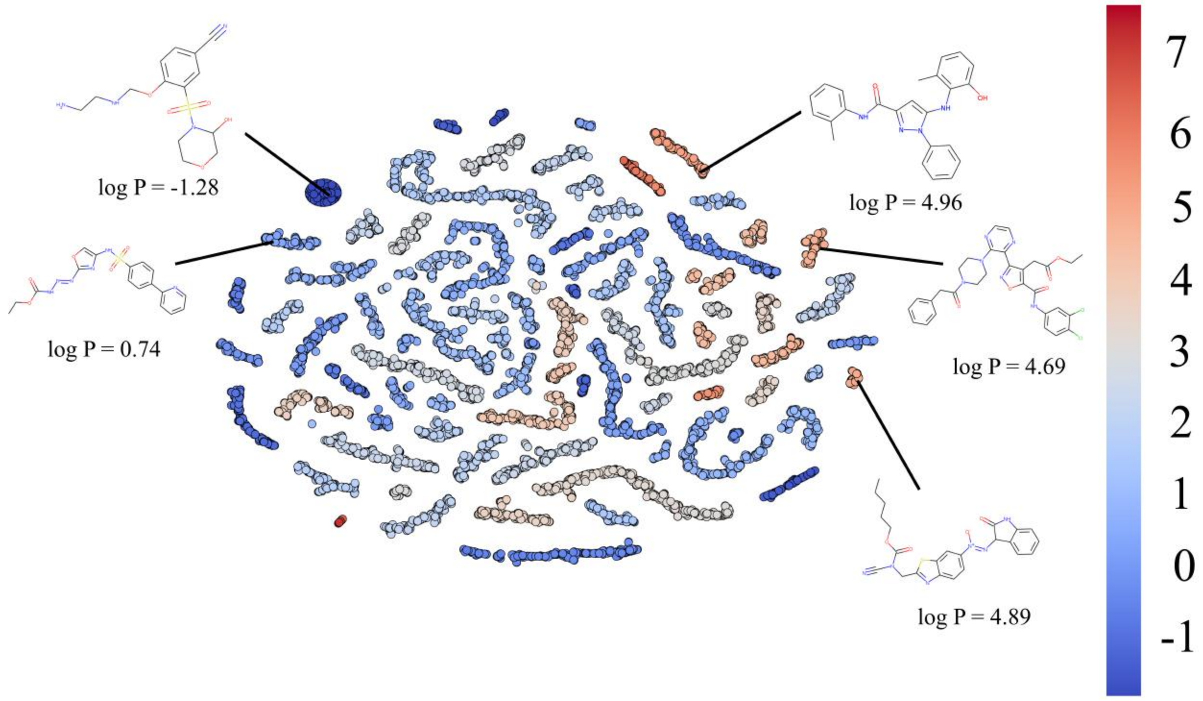


Рис. 4-3: t-SNE diagram for partition coefficient

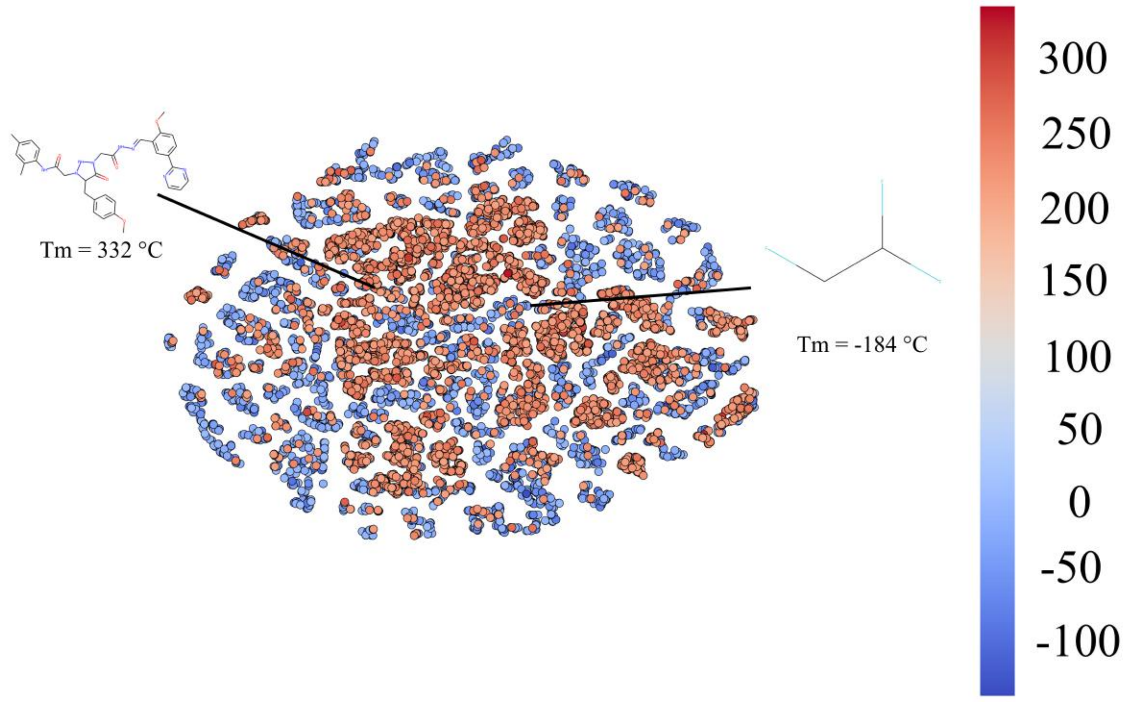


Рис. 4-4: t-SNE diagram for melting temperature

Глава 5

Заключение

In this thesis we proposed end-to-end deep learning system for *de novo* molecular design. The system is data-driven and does not rely on hand-crafted features. Deep learning is poised to transform drug discovery and computational chemistry. As a pilot application, we systematically demonstrate how Deep Reinforcement Learning model can be used for *de novo* computational drug design. In experimental part we showed how Deep RL system can generate chemically sensible molecules with optimized properties. We took into consideration five different properties, which include physical, chemical, bio-activity and structural one. To out best knowledge, this is a first case of optimizing bio-activity property, described in scientific literature. Further development of the system includes overcoming existing limitations such as predictive model weaknesses exploitation and also extending the system for optimizing several properties simultaneously.

Литература

[1] Jack W Scannell, Alex Blanckley, Helen Boldon, and Brian Warrington. Diagnosing the decline in pharmaceutical R&D efficiency. *Nature reviews. Drug discovery*, 11(3):191–200, 2012.

[2] Artificial intelligence: The return of the machinery question. accessed Feb 23, 2017.

[3] Saurabh Jha and Eric J Topol. Adapting to artificial intelligence: radiologists and pathologists as information specialists. *JAMA*, 316(22):2353–2354, 2016.

[4] Katie Chockley and Ezekiel Emanuel. The end of radiology? three threats to the future practice of radiology. *Journal of the American College of Radiology*, 13(12):1415–1420, 2016.

[5] Matthew Ragoza, Joshua Hochuli, Elisa Idrobo, Jocelyn Sunseri, and David Ryan Koes. Protein-ligand scoring with convolutional neural networks. *arXiv preprint arXiv:1612.02751*, 2016.

[6] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *arXiv preprint arXiv:1611.03199*, 2016.

[7] Marwin HS Segler and Mark P Waller. Modelling chemical reasoning to predict and invent reactions. *Chemistry-A European Journal*, 2017.

[8] Justin S Smith, Olexandr Isayev, and Adrian E Roitberg. Ani-1: An extensible neural network potential with dft accuracy at force field computational cost. *arXiv preprint arXiv:1610.08935*, 2016.

[9] Volker Schnecke and Jonas Boström. Computational chemistry-driven decision making in lead generation. *Drug discovery today*, 11(1):43–50, 2006.

[10] Ricardo Macarron. Critical review of the role of hts in drug discovery. *Drug discovery today*, 11(7):277–279, 2006.

[11] Gisbert Schneider and Uli Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663, 2005.

[12] Harald Mauser and Wolfgang Guba. Recent developments in de novo design and scaffold hopping. *Current opinion in drug discovery & development*, 11(3):365–374, 2008.

[13] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

[14] Pavel G Polishchuk, TI Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679, 2013.

[15] Christopher Lipinski and Andrew Hopkins. Navigating chemical space for biology and medicine. *Nature*, 432(7019):855–861, 2004.

[16] Daniel Reker and Gisbert Schneider. Active-learning strategies in computer-assisted drug discovery. *Drug discovery today*, 20(4):458–465, 2015.

[17] Petra Schneider and Gisbert Schneider. De novo design at the edge of chaos: Miniperspective. *Journal of medicinal chemistry*, 59(9):4077–4086, 2016.

[18] Nathan Brown, Ben McKay, François Gilardoni, and Johann Gasteiger. A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. *Journal of chemical information and computer sciences*, 44(3):1079–1087, 2004.

[19] Rafael Gómez-Bombarelli, David Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *arXiv preprint arXiv:1610.02415*, 2016.

[20] Kristopher De Asis, J Fernando Hernandez-Garcia, G Zacharias Holland, and Richard S Sutton. Multi-step reinforcement learning: A unifying algorithm. *arXiv preprint arXiv:1703.01327*, 2017.

[21] Marina Krakovsky. Reinforcement renaissance. *Communications of the ACM*, 59(8):12–14, 2016.

[22] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[23] H Jaap Van Den Herik, Jos WHM Uiterwijk, and Jack Van Rijswijck. Games solved: Now and in the future. *Artificial Intelligence*, 134(1-2):277–311, 2002.

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. *Advances in Neural Information Processing Systems 27*, pages 2672–2680, 2014.

[25] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv*, 2016.

[26] James Martens. Generating Text with Recurrent Neural Networks. *Neural Networks*, 131(1):1017–1024, 2011.

[27] David Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, 1988.

[28] Armand Joulin and Tomas Mikolov. Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets. *arXiv*, pages 1–10, 2015.

[29] Tristan Deleu and Joseph Dureau. Learning Operations on a Stack with Neural Turing Machines. *arXiv*, pages 1–6, 2016.

[30] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1828–1836, 2015.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *Nips*, pages 1–9, 2013.

[32] Sepp Hochreiter and Jurgen Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1–32, 1997.

[33] Ronald J. Willia. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256, 1992.

[34] A. Patrı́cia Bento, Anna Gaulton, Anne Hersey, Louisa J. Bellis, Jon Chambers, Mark Davies, Felix A. Krüger, Yvonne Light, Lora Mak, Shaun McGlinchey, Michal Nowotka, George Papadatos, Rita Santos, and John P. Overington. The ChEMBL bioactivity database: An update. *Nucleic Acids Research*, 42(D1), 2014.

[35] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*, pages 1–9, 2014.

[36] ChemAxon. Marvin Sketch, 2013.

[37] John J. Irwin and Brian K. Shoichet. ZINC - A free database of commercially available compounds for virtual screening. *Journal of Chemical Information and Modeling*, 45(1):177–182, 2005.

[38] L J P Van Der Maaten and G E Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.