Li *et al. J Cheminform (2018) 10:33*
https://doi.org/10.1186/s13321-018-0287-6

Journal of Cheminformatics

RESEARCH ARTICLE

Open Access

Multi-objective de novo drug design with conditional graph generative model

Yibo Li, Liangren Zhang* and Zhenming Liu*

Abstract

Recently, deep generative models have revealed itself as a promising way of performing de novo molecule design. However, previous research has focused mainly on generating SMILES strings instead of molecular graphs. Although available, current graph generative models are are often too general and computationally expensive. In this work, a new de novo molecular design framework is proposed based on a type of sequential graph generators that do not use atom level recurrent units. Compared with previous graph generative models, the proposed method is much more tuned for molecule generation and has been scaled up to cover significantly larger molecules in the ChEMBL database. It is shown that the graph-based model outperforms SMILES based models in a variety of metrics, especially in the rate of valid outputs. For the application of drug design tasks, conditional graph generative model is employed. This method offers highe flexibility and is suitable for generation based on multiple objectives. The results have demonstrated that this approach can be effectively applied to solve several drug design problems, including the generation of compounds containing a given scaffold, compounds with specific drug-likeness and synthetic accessibility requirements, as well as dual inhibitors against JNK3 and GSK-3β.

Keywords: Deep learning, De novo drug design, Graph generative model

Background

The ultimate goal of drug design is the discovery of new chemical entities with desirable pharmacological properties. Achieving this goal requires medicinal chemists to explore the chemical space for new molecules, which is proved to be extremely difficult, mainly due to the size and complexity of the chemical space. It is estimated that there are around $10^{60}-10^{100}$ synthetically available molecules [1]. Meanwhile, the space of chemical compounds exhibits a discontinues structure, making searching difficult to perform [2].

De novo molecular design aims at assisting this processes with computer-based methods. Early works have developed various algorithms to produce new molecular structures, such as atom based elongation or fragment based combination [3, 4]. Those algorithms are often coupled with global optimization techniques such as ant colony optimization [5, 6], genetic algorithms [7, 8] or particle swam optimization [9] for the generation of molecules with desired properties.

Recent developments in deep learning [10] have shed new light on the area of de novo molecule generation. Previous works have shown that deep generative models are very effective in modeling the SMILES representation of molecules using recurrent neural networks (RNN), an architecture that has been extensively applied to tasks related sequential data [11]. Segler et al. [12] applied SMILES language model (LM) on the task of generating focused molecule libraries by fine-tuning the trained network with a smaller set of molecules with desirable properties. Olivecrona et al. [13] used a GRU [14] based LM trained on the ChEMBL [15] dataset to generate SMILES string. The mode is then fine-tuned using reinforcement learning for the generation of molecules with specific requirements. Popova et al. [16] proposed to integrate the generative and predictive network together in the generation phase. Beside language model, Gómez–Bombarelli et al. [13] used variational autoencoder (VAE) [17] to generate drug-like compounds from ZINC database [18].

*Correspondence: liangren@bjmu.edu.cn; zmliu@bjmu.edu.cn
State Key Laboratory of Natural and Biomimetic Drugs, School of Pharmaceutical Sciences, Peking University, Xueyuan Road 38, Haidian District, Beijing 100191, China

This work aimed at obtaining a bi-directional mapping between molecule space and a continuous latent space so that operations on molecules can be achieved by manipulating the latent representation. Blaschke et al. [19] compared different architectures for VAE and applied it to the task of designing active compounds against DRD2.

The researches described above demonstrated the effectiveness of SMILES based model regarding molecule generation. However, producing valid SMILES strings requires the model to learn rules that are irrelevant to molecular structures, such as the SMILES grammar and atom ordering, which increases the burden of the model and makes the SMILES string a less preferable representation compared with molecular graphs. Research in deep learning has recently enabled the direct generation of molecular graphs. Johnson et al. [20] proposed a sequential generation approach for graphs. Though their implementation is mainly for reasoning tasks, this framework is potentially applicable to molecule generation. A more recent method [21] was proposed for generating the entire graph all at once. This model has been successfully applied to the generation of small molecular graphs. The implementation that is most similar to ours is by the recent work by Li et al. [22] using a sequential decoding scheme similar to that by Johnson et al. Decoding invariance is introduced by sampling different atom ordering from a predefined distribution. This method has been applied to the generation of molecules with less than 20 heavy atoms from ChEMBL dataset. Though inspiring, the methods discussed above have a few common problems. First of all, the generators proposed are relatively general. This design allows those techniques to be applied to various scenarios but requires further optimization for application in molecule generation. Secondly, many of those models suffer from scalability issue, which restricts the application to molecules with small sizes.

In this work, we propose a graph-based generator that is more suited for molecules. The model is scaled to cover compounds containing up to 50 heavy atoms in the ChEMBL dataset. Results show the graph-based model proposed is able to outperform SMILES based methods in a variety metrics, including the rate of valid outputs, KL and JS divergence of molecular properties, as well as NLL loss. A conditional version of the model is employed to solve various drug design related tasks with multiple objectives, and results have demonstrated promising performance.

Methods

Molecular graph

Molecular graph is a way of representating the structural information of molecules using graphs ($G = (V, E)$). Atoms and bonds in the molecule are viewed as graph nodes ($v \in V$) and edges ($e \in E$). Each node is labeled with its corresponding atom type, while each edge is labeled with its corresponding bond type. We refer the set of all atom types and bond types as A and B respectively. In this work, the atom type is specified using three variables: the atomic symbol (or equally the atomic number), the number of explicit hydrogens attached, and the number of formal charges. For example, the nitrogen atom in pyrrole can be represented as the triple ("N", 1, 0). The set of all atom types (A) is extracted from molecules in the ChEMBL dataset (see Additional file 1: Supplementary Text 1), and contains 33 members in total. For bonds, we only consider the following four bond types: single, double, triple and aromatic. A visualized demonstration of molecular graph is given in Fig. 1.

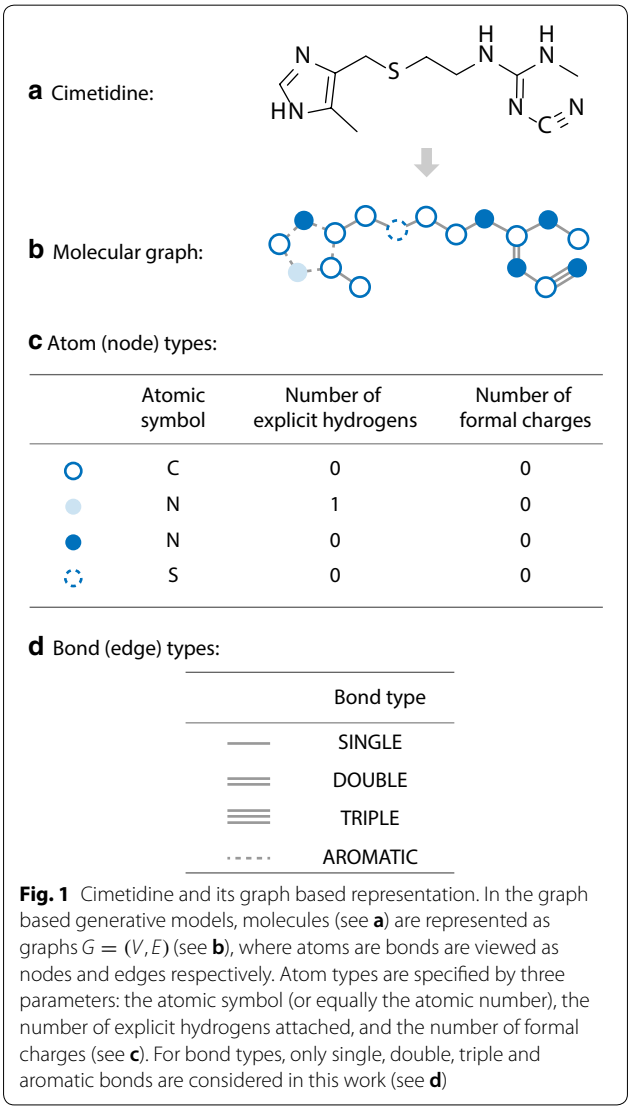

Fig. 1 Cimetidine and its graph based representation. In the graph based generative models, molecules (see **a**) are represented as graphs $G = (V, E)$ (see **b**), where atoms are bonds are viewed as nodes and edges respectively. Atom types are specified by three parameters: the atomic symbol (or equally the atomic number), the number of explicit hydrogens attached, and the number of formal charges (see **c**). For bond types, only single, double, triple and aromatic bonds are considered in this work (see **d**)

Graph generative model

We now consider the deep generative models that can directly output molecular graphs. In this work, we mainly focus on sequential graph generators, which build graph by iteratively refining its intermediate structure. The process starts from the empty graph $G_0 = (\emptyset, \emptyset)$. At step i, a graph transition t_i is selected from the set of all available transition actions $T(G_i)$ based on the generation history $(G_0, \ldots, G_i)$. The selection is done by sampling t_i from a probability distribution $t_i \sim p_{\theta}(t_i|G_i, \ldots, G_0)$, which is parametrized by a deep network. Then, t_i is performed on G_i to get the graph structure for the next step $G_{i+1} = t_i(G_i)$. At the final step n, termination operation t^* is performed and the model outputs $G = G_n$ as the final result.

The entire process is illustrated in Fig. 2. We call the mapping T, which determines all available graph transitions at each step, a *decoding scheme*. The sequence $r = ((G_0, t_0), (G_1, t_1), \ldots, (G_n, t_n))$ is called a *decoding route* of G, and the distribution $p_{\theta}(t_i|G_i, \ldots, G_0)$ is called a *decoding policy*.

Previous graph generative models are usually too general and less optimized for the generation of molecular graphs. Here we offer the following optimizations:

1. A much simpler decoding scheme T is used to decrease the number of steps required for generation.
2. No atom level recurrent unit is used in the decoding policy. Instead, we explored two other options: (1) parametrizing the decoding policy as a Markov process and (2) using only molecule level recurrent unit. Those modifications helps to increase the scalability of the model.
3. During the calculation of log-likelihood loss, we sample r from a parametrized distribution $q_{\alpha}(r|G)$. The parameter α controls the degree of randomness of q_{α}, offering higher flexibility for the model.

The following three sections are devoted to the detailed discussions of the optimizations above.

Decoding scheme

The transitions in $T(G_i)$ given the intermediate state G_i is restricted to the following four types:

1. *Initialization* At the beginning of the generation, the only allowed transition is to add the first atom to the empty graph G_0.

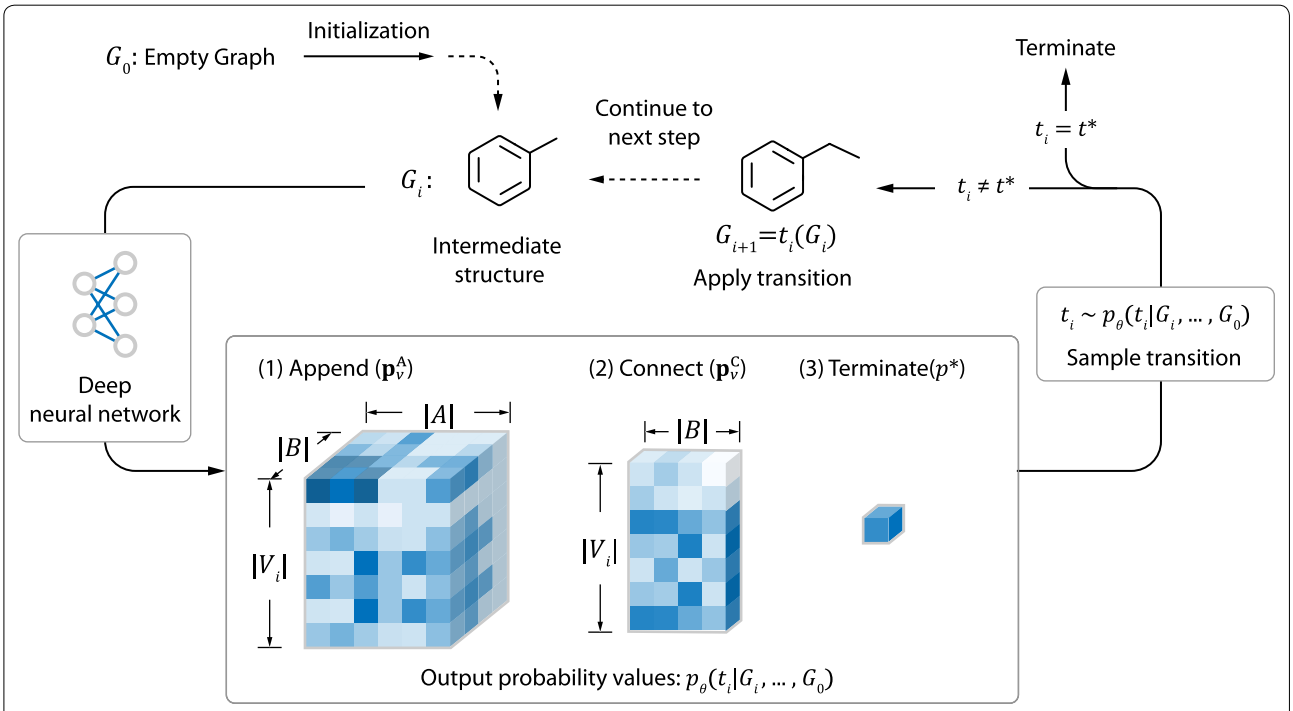

Fig. 2 A schematic representation of molecule generation process. Starting with the empty graph G_0, initialization is performed to add the first atom. At each step, a graph transition (append, connect or terminate) is sampled and performed on the intermediate molecule structure. The probability for sampling each transition is given by $p_{\theta}(t|G_i, \ldots, G_0)$, which is parametrized using deep neural network. Finally, termination operation is performed to end the generation

2. *Append* This action adds a new atom to G_i and connect it to an existing atom with a new bond.
3. *Connect* This action connects two existing atoms $v_1, v_2 \in V_i$ with a new bond. For simplicity, we only allow connections to start from the latest appended atom v^*, which means that $v_1 = v^*$.
4. *Termination* (t^*) End the generation process.

The entire process is shown in Fig. 2, and a more detailed illustration is provided in Additional file 2: Figure S1 and S2. In theory, $T(G)$ should not contain actions that violate the chemical validity of molecules. However, in order to test the ability for the model to learn those constraints, we do not explicity exclude those actions from $T(G)$ during training.

Note that compared with the implementation in [22], the action of adding new atom and the action of connecting it to the molecule is merged into a single "append" step. This helps to reduce the number of steps during generation. It is easy to show that the number of steps required for generating graph $G = (V, E)$ equals exactly to $|E| + 2$, which is generally much smaller than the length of the corresponding SMILES string (as shown in Additional file 2: Figure S3).

Decoding policy

During generation, the decoding policy p_{θ} need to specify the probability value for each graph transition in $T(G_i)$. More specifically, p_{θ} need to output the following probability values:

1. $\mathbf{p}_v^{\mathrm{A}}$ *for each* $v \in V_i$ A matrix with size $|A| \times |B|$, whose element $(\mathbf{p}_v)_{ab}$ represents the probability of appending a new atom of type $a \in A$ to atom v with a new bond of type $b \in B$.
2. $\mathbf{p}_v^{\mathrm{C}}$ *for each* $v \in V_i$ A vector with size $|B|$, whose element $(\mathbf{p}_v^{\mathrm{C}})_b$ represents the probability of connecting the latest added atom v^* with v using a new bond of type $b \in B$.
3. p^* A scalar value indicating the probability of terminating the generation.

A visualized depiction of $\mathbf{p}_v^{\mathrm{A}}$, $\mathbf{p}_v^{\mathrm{C}}$ and p^* is shown in Fig. 2. The decoding policy p_{θ} is parameterized using neural network. At each step, the network accepts the the decoding history $(G_0, \ldots, G_i)$ as input and calculates the probability values $(\mathbf{p}_v^{\mathrm{A}}, \mathbf{p}_v^{\mathrm{C}}, p^*)$ as output. In this work, we explored two novel graph generation architectures, namely MolMP and MolRNN. Unlike the methods proposed in [20, 22], the two architectures do not involve atom level recurrency, which helps to increase the scalability of the model.

MolMP

MolMP models graph generation as a Markov process, where the transition of G_i only depends on the current state of the graph, not on the history (Fig. 3a). This means that $p_{\theta}(t|G_i, \ldots, G_0) = p_{\theta}(t|G_i)$. Since this type of architecture does not include any recurrent units, it will be less expensive compared with RNN based models. Moreover, the computation at different steps can be easily parallelized during training. The detailed architecture of MolMP is given as follows:

1. An initial atom embedding $\mathbf{h}_v^0$ is first generated for each atom v:

$$\mathbf{h}_v^0 = \mathrm{Embedding}_{\theta}(v) \tag{1}$$

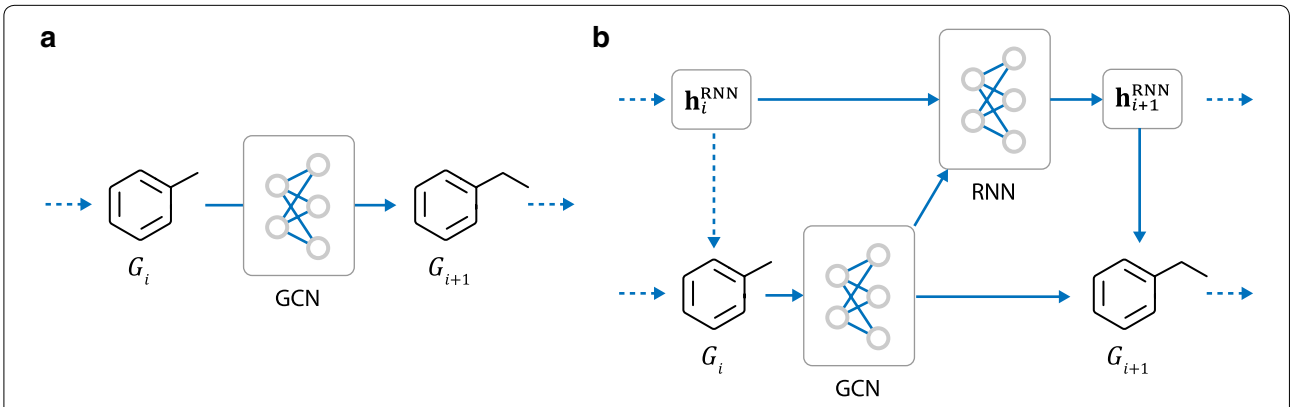

Fig. 3 The two type of graph generative architectures explored in this work: **a** MolMP: this architecture treats graph generation as a Markov process, in which the transition of G_i only depends on the current state of the graph, not on the history. **b** MolRNN: this architecture adds a single molecule level recurrent unit to MolMP

$\mathbf{h}_v^0$ is determined based on the following information: (1) the atom type of v and (2) whether v is the latest appended atom. The dimension of $\mathbf{h}_v^0$ is set to 16.

2. $\mathbf{h}_v^0$ is passed to a sequence of L graph convolutional layers:

$$\mathbf{h}_v^l = \text{GraphConv}_{\boldsymbol{\theta}}^l(\mathbf{h}_v^{l-1}, G_i) \tag{2}$$

where $l = 1, \ldots, L$. Except the first layer, each convolutional layer $\text{GraphConv}_{\boldsymbol{\theta}}^l$ adopts a "BN-ReLU-Conv" structure as suggested in [23]. The detailed architecture of graph convolution is described in "Graph Convolution". We use six convolution layers in this work ($L = 6$), each with 32, 64, 128, 128, 256, 256 output units.

The outputs from all graph convolutional layers are then concatenated together, followed by batch normalization and ReLU:

$$\mathbf{h}_v^{\text{skip}} = relu\left(bn(\text{Concat}(\mathbf{h}_v^1, \ldots, \mathbf{h}_v^L))\right) \tag{3}$$

3. $\mathbf{h}_v^{\text{skip}}$ is passed to the fully connected network $\text{MLP}_{\boldsymbol{\theta}}^{\text{FC}}$ to obtain the final atom level representation $\mathbf{h}_v$.

$$\mathbf{h}_v = \text{MLP}_{\boldsymbol{\theta}}^{\text{FC}}(\mathbf{h}_v^{\text{skip}}) \tag{4}$$

$\text{MLP}_{\boldsymbol{\theta}}^{\text{FC}}$ consists of two linear layers, with 256 and 512 output units each. Batch normalization and ReLU are applied after each layer.

4. Average pooling is applied at graph level to obtain the molecule representation $\mathbf{h}_{G_i}$:

$$\mathbf{h}_{G_i} = AvgPool([\mathbf{h}_v]_{v \in V_i}) \tag{5}$$

5. The probability value for each action is produced by first calculate the unnormalized values ($\hat{\mathbf{p}}_v^A$, $\hat{\mathbf{p}}_v^C$ and $\hat{p}^*$) as follows:

$$\left[\hat{\mathbf{p}}_v^A, \hat{\mathbf{p}}_v^C\right] = \text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}_v, \mathbf{h}_{G_i}) \tag{6}$$

$$\hat{p}^* = \text{MLP}_{\boldsymbol{\theta}}^*(\mathbf{h}_{G_i}) \tag{7}$$

Those values are then normalized to get the final result:

$$\mathbf{p}_v^{\text{A}} = \hat{\mathbf{p}}_v^A / P \tag{8}$$

$$\mathbf{p}_v^{\text{C}} = \hat{\mathbf{p}}_v^C / P \tag{9}$$

$$p^* = \hat{p}^* / P \tag{10}$$

where $P = \sum_{vab}(\hat{\mathbf{p}}_v^{\text{A}})_{ab} + \sum_{vb}(\hat{\mathbf{p}}_v^{\text{C}})_b + \hat{p}^*$

$\text{MLP}_{\boldsymbol{\theta}}$is a two layer fully connected network with hidden size 128 and output size $|A| \times |B| + |B|$. This output is then split into the matrix $\hat{\mathbf{p}}_v^A$ of size $|A| \times |B|$ and the vector $\hat{\mathbf{p}}_v^C$ of length $|B|$. MLP* is a one layer fully connected network. Both $\text{MLP}_{\boldsymbol{\theta}}$ and MLP* uses exponential activiaton in the output layer.

The architecture of the entire network is shown in Fig. 4.

MolRNN

The second architecture adds a single molecule level recurrent unit to MolMP, as shown in Fig. 3. We refer

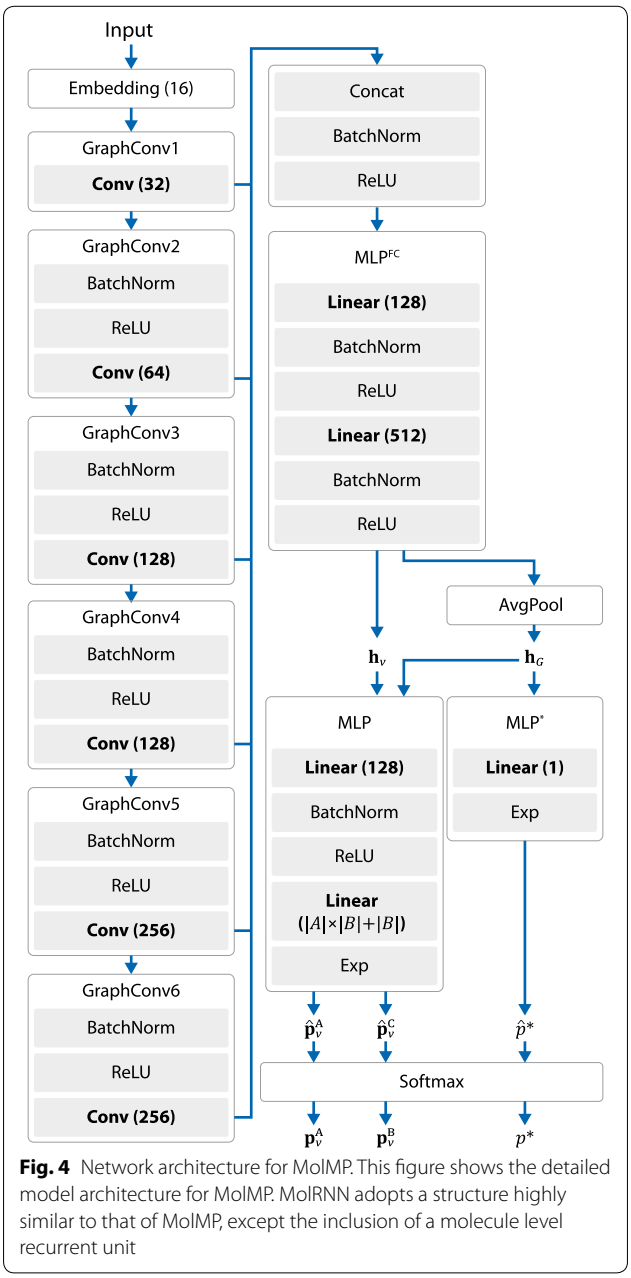

Fig. 4 Network architecture for MolMP. This figure shows the detailed model architecture for MolMP. MolRNN adopts a structure highly similar to that of MolMP, except the inclusion of a molecule level recurrent unit

to this method as MolRNN. The model architecture is specified as follows:

1. First of all, the model generates the atom level ($\mathbf{h}_v, v \in V_i$) and molecule level ($\mathbf{h}_{G_i}$) representation for the graph state G_i. This part of the network uses the same architecture as that in MolMP.
2. Given $\mathbf{h}_v$ and $\mathbf{h}_{G_i}$, the hidden state of the molecule level recurrent unit ($\mathbf{h}_i^{RNN}$) is updated as:

$$\mathbf{h}_{i+1}^{\mathrm{RNN}} = \mathrm{RNN}_{\boldsymbol{\theta}}(\mathbf{h}_i^{\mathrm{RNN}}, \mathbf{h}_{v*}, \mathbf{h}_{G_i}) \tag{11}$$

   where $\mathbf{h}_{v*}$ is the representation of the latest appended atom v^*. The recurrent network $\mathrm{RNN}_{\boldsymbol{\theta}}$ is employed using three GRU layers with a hidden size of 512.
3. The probability values $\mathbf{p}_v^{\mathrm{A}}$, $\mathbf{p}_v^{\mathrm{C}}$, p^* are calculated in the same manner as MolMP by replacing $\mathbf{h}_{G_i}$ in Eqs. 6 and 7 with $\mathbf{h}_{i+1}^{\mathrm{RNN}}$.

The overall architecture of MolRNN is highly similar to that of MolMP. However, it is found that the molecule level recurrent unit in MolRNN provides significant improvements to the model performance (see "Model performance and sample quality"), while inducing little extra computational cost compared with MolMP.

Graph convolution

In this work, we rely on graph convolutional network (GCN) [24] to extract information from intermediate graph states G_i. Each graph convolutional layer adopts the "BN-ReLU-Conv" structure as described before. In terms of the convolution part, the architecture is structured as follows:

$$\begin{aligned}\mathbf{h}_v^l = \mathbf{W}^l \mathbf{h}_v^{l-1} &+ \sum_{b \in B} \Theta_b^l \sum_{u \in N_b^{bond}(v)} \mathbf{h}_u^{l-1} \\ &+ \sum_{1<d\le D} \Phi_d^l \sum_{u \in N_d^{path}(v)} \mathbf{h}_u^{l-1}\end{aligned} \tag{12}$$

where $\mathbf{h}_v^l$ is the output representation of atom v at layer l, and $\mathbf{h}_v^{l-1}$ is the input representation. $N_b^{bond}(v)$ is the set of all atoms directly connected to atom v with bond of type b, and $N_d^{path}(v)$ is the set of all atoms whose distance to atom v equals to d. D represents the receptive field size, which is set to 3 in this work. W^l, Θ_b^l and Φ_d^l are weight parameters of layer l.

In brief, the output representation of atom v at each layer l ($\mathbf{h}_v^l$) is calculated according to the following information:

1. The input representation of v ($\mathbf{h}_v^{l-1}$),
2. Information of local neighbors, which is given by $\sum_{b\in B} \Theta_b^l \sum_{u\in N_b^{bond}(v)} \mathbf{h}_u^{l-1}$. Note that this part of information is conditioned on the bond type b between v and its neighborhood atom u.
3. Information of remote neighbors, given by $\sum_{1<d\le D} \Phi_d^l \sum_{u\in N_d^{path}(v)} \mathbf{h}_u^{l-1}$. This part of information is conditioned on the distance d between v and its remote neighbor u.

The architecture is illustrated in Fig. 5.

Our implementation of graph convolution is similar to the edge conditioned convolution by Simonovsky et al. [25], except that we directly include the information of remote neighbors of v in order to achieve a larger receptive field with fewer layers.

Likelihood function

To train the generative model, we need to maximize the log-likelihood $p_{\boldsymbol{\theta}}(G)$ for the training samples. However, for the step-wise generative models discussed above, the likelihood is only tractable for a given decoding route $r = ((G_0, t_0), (G_1, t_1), \ldots, (G_n, t_n))$:

$$\log p_{\boldsymbol{\theta}}(G, r) = \sum_{i=0}^{n} \log p_{\boldsymbol{\theta}}(t_i | G_i, \ldots, G_0) \tag{13}$$

While the marginal likelihood can be computed as:

$$\log p_{\boldsymbol{\theta}}(G) = \log \sum_{r \in R(G)} p_{\boldsymbol{\theta}}(G, r) \tag{14}$$

where $R(G)$ is the set of all possible decoding route for G. The marginal likelihood function is intractable for most molecules encountered in drug design. One way to resolve this problem is to use importance sampling as proposed in [22]:

$$\log p_{\boldsymbol{\theta}}(G) = \log \mathbb{E}_{r\sim q(r|G)} \left[\frac{p_{\boldsymbol{\theta}}(G, r)}{q(r|G)} \right] \tag{15}$$

where $q(r|G)$ is a predefined distribution on $R(G)$. Both the deterministic and the fully randomized $q(r|G)$ were explored in the previous work [22]. However, a more desirable solution would lie in somewhere between deterministic decoding and fully randomized decoding. In this work, instead of sample from the distribution $q(r|G)$, we sample r from distribution $q_\alpha(r|G)$ that is parameterized by $0 \le \alpha \le 1$. $q_\alpha(r|G)$ is designed such that the decoding will largely follow depth first decoding with canonical

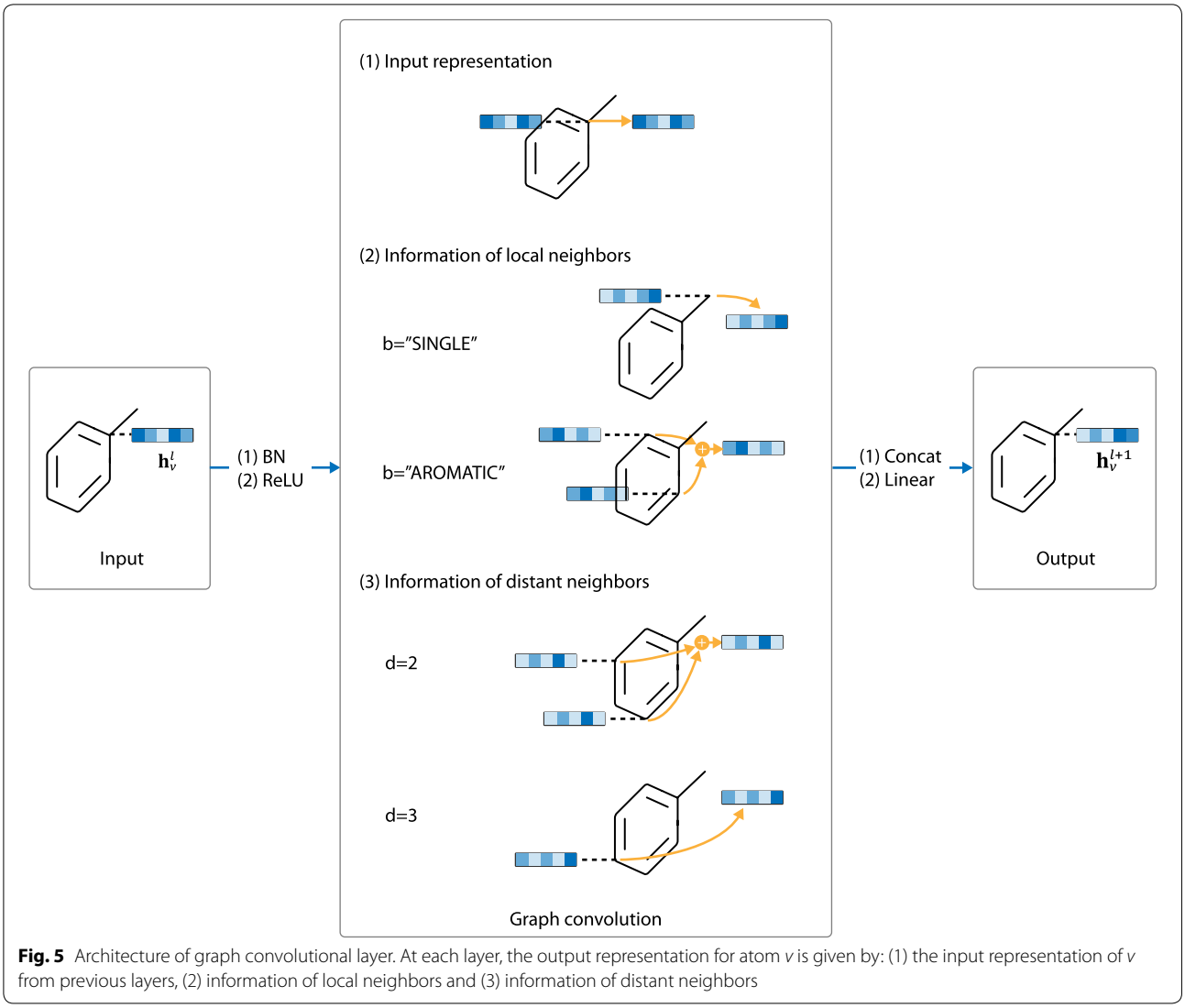

Fig. 5 Architecture of graph convolutional layer. At each layer, the output representation for atom *v* is given by: (1) the input representation of *v* from previous layers, (2) information of local neighbors and (3) information of distant neighbors

ordering, but at each step, there is a small possibility $1-\alpha$ that the model will make a random mistake. In this way, the parameter α measures can be used to control the randomness of the distribution q_α. The algorithm is shown in Additional file 1: Supplementary Text 4.

$$\begin{aligned}\log p_{\boldsymbol{\theta}}(G) &= \log \mathbb{E}_{r\sim q_\alpha(r|G)}\left[\frac{p_{\boldsymbol{\theta}}(G,r)}{q_\alpha(r|G)}\right] \\ &\geq \log \frac{1}{k}\sum_{i=1}^{k}\frac{p_{\boldsymbol{\theta}}(G,r_i)}{q_\alpha(r_i|G)}\end{aligned} \tag{16}$$

For $\alpha = 1$, the distribution falls back to the deterministic decoding. The parameter α is treated as a hyperparameter which is optimized for model performance. We tried $\alpha \in \{1.0, 0.8, 0.6\}$ on both MolMP and MolRNN.

Conditional generative model

Most molecule design tasks require to produce compounds satisfying certain criteria, such as being synthetically available or having a high affinity for a certain target. Previous researches have developed various methods to achieve objective directed molecule generation. Segler et al. [12] used transfer learning in the design of focused compound libraries. Olivecrona et al. [13] applied reinforcement learning (RL) in the objective based chemical design and have reported promising performance in various tasks. Guimaraes et al. [26] proposed ORGAN, which combines SeqGAN with an domain-specific objective term, and showed that ORGAN is effective in the optimization of different molecular properties. Neil et al. [27] created a benchmark analysis of various RL based method in different tasks of molecule design. In this work, we explored another way to achieve requirement

based molecule design using conditional generative model. We first convert the given requirement to the numerial representation called conditional code (**c**), and the generative model is then modified to be conditioned on **c**. For graph generative model, this means that the decoding policy is now $p_{\theta}(t_i|G_i,\ldots,G_0,\mathbf{c})$ (see Fig. 6). Compared with previous approaches by Olivecrona et al. and Guimaraes et al., conditional generative model does not require reinforcement learning, and provides the following flexibilities:

1. The conditional code can incorporate both discrete and continuous objectives, and can easily scale to multiple objective.
2. When changing the generation objective, previous methods usually require the model to be retained on the new condition. But for conditional generative models, this can be achieve simply by changing the conditional code input c.

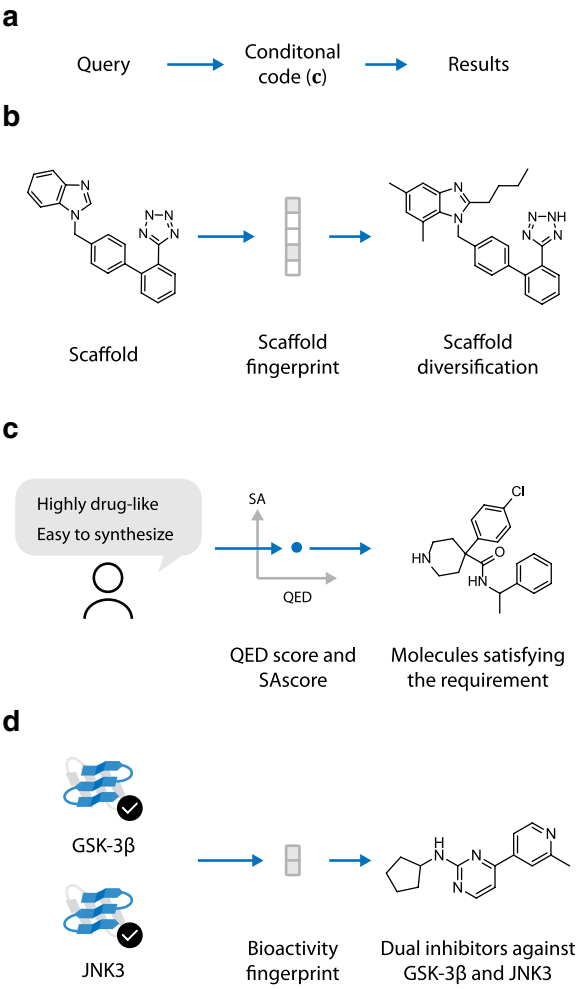

Fig. 6 Conditonal generative models: **a** For the generation of molecules based on a given requriement, the requriement (query) is first converted to the numerical representation called conditoinal code, the generative model is then modified to be conditioned on the conditonal code. **b** Scaffold based molecule generation: The model is required to generate molecules based on a given scaffold. This requriement is first converted to the conditional code called scaffold fingerprint. Then the molecule containing the query scaffold is generated based on the fingerprint. **c** Generation based on drug-likeness and synthetic accessibility: Drug-likeness and synthetic accessibility are first quantized using QED and SAscore. Then model then generates molecules based on the two properties. **d** Designing of dual inhibitors of JNK3 and GSK-3β: the generation process is based on the bioactivity fingerprint containing the activity requirement of each target

Both graph based and SMILES based conditional generators are implemented in this work. For graph based model, the graph convolution is modified to include **c** as input:

$$\begin{aligned}\mathbf{h}_v^l = &\mathbf{W}^l\mathbf{h}_v^{l-1} \\ &+\sum_{b\in B}\Theta_b^l\sum_{u\in N_b^{bond}(v)}\mathbf{h}_u^{l-1} \\ &+\sum_{1<d\le D}\Phi_d^l\sum_{u\in N_d^{path}(v)}\mathbf{h}_u^{l-1}+\Psi^l\mathbf{c}\end{aligned} \tag{17}$$

Simply state, **c** is included in the graph convoludion architecture by adding an additional term $\Psi^l\mathbf{c}$ to the unconditional implementation in Eq. 12. For SMILELS based model, the conditional code is included by concatenating it with the input at each step: $\mathbf{x}_i' = \text{Concat}(\mathbf{x}_i, \mathbf{c})$. where $\mathbf{x}_i$ is the one-hot representation of the SMILES charactor input at step i.

Conditional models have already been used by the previous work [21] for molecule generation, but was restricted to simple properties such as the number of heavy atoms as conditional codes. Also, the method have not yet applied to multi-objective molecule generation. Here, we apply this method to other more complexed drug design tasks, including scaffold-based generation, property-based generation and the design of dual inhibitor of JNK3 and GSK-3β (see 6). The best performing graph and SMILES based generator (see "Model performance and sample quality") are implemented in conditionalized version and applied to those tasks.

Scaffold-based generation

The concept of molecular scaffold has long been of significant importance in medicinal chemistry [28]. Though various definitions are available, the most widely accepted definition is given by Bemis and Murcko [29], who proposed derive the scaffold of a given molecule by removing all side chain atoms. Studies have found various scaffolds that have privileged characteristics in terms of the activity of certain target [30–32]. Once such

privileged structure is found, a related task is to produce compound libraries containing such scaffolds for subsequent screening.

Here, conditional graph generative model is applied to generate compounds containing scaffold s, which is drawn from the pre-defined scaffold set $S = \{s_i\}_{i=1}^{N_S}$. The set S is extracted from the list of approved drugs in DrugBank [33]. Two types of structures are extracted from the molecules to construct S: (1) the Bemis–Murcko scaffolds, and (2) ring assemblies. Ring assemblies are included in S since we found that including extra structural information beside Bemis–Murcko scaffolds helps to improve the conditional generation performance. Detailed scaffold extraction workflow is shown in Additional file 1: Supplementary Text 2. For each molecule G, the conditional code $\mathbf{c} = (c_1, c_2, \ldots, c_{N_S})$ is set to be the binary vector such that $c_i = 1$ if G contains s_i as substructure, and $c_i = 0$ otherwise. We refer $\mathbf{c}$ as the scaffold fingerprint of G, as it can in fact be viewed as a substructure fingerprint based on scaffold set S. To generate molecule containing substructure $s \in S$, the fingerprint $\mathbf{c}_s$ for s is used as conditional code. The output should contain two type of molecules:

1. Molecules containing s as its Bemis–Murcko scaffold.
2. Molecules whose Bemis–Murcko scaffold contains s but does not reside inside S.

The procedure is better demonstrated in Fig. 7.

Using this method, detailed control can be performed on the scaffold of the output structure.

Generation based on synthetic accessibility and drug-likeness

Drug-likeness and synthetic accessibility are two properties that have significant importance in the development of novel drug candidate. Drug-likeness measures the consistency of a given compound with the currently known drugs in terms of the structural or physical properties, and is frequently used to filter out obvious non-drug like compounds in the early phase of screening [34, 35]. Synthetic accessibility is also an important property for de novo drug design since subsequent experimental validation requires synthesis of the given compound [36]. In this task, the model is required to generate molecules according to a given level of drug-likeness and synthetic accessibility. The drug-likeness is measured using the Quantitative Estimate of Drug-likeness (QED) [37], and synthetic accessibility is evaluated using the SA score [36]. The conditional code $\mathbf{c}$ is defined as $\mathbf{c} = (QED, SA)$, where the QED and SA score is all calculated using RDKit [38].

In practice, instead of specifying a single value of QED and SA score, we often use intervals to express the requirements for desired output molecules. This means that we are required to sample molecules from the distribution $p_{\theta}(G|\mathbf{c} \in C)$, where the generation requirement

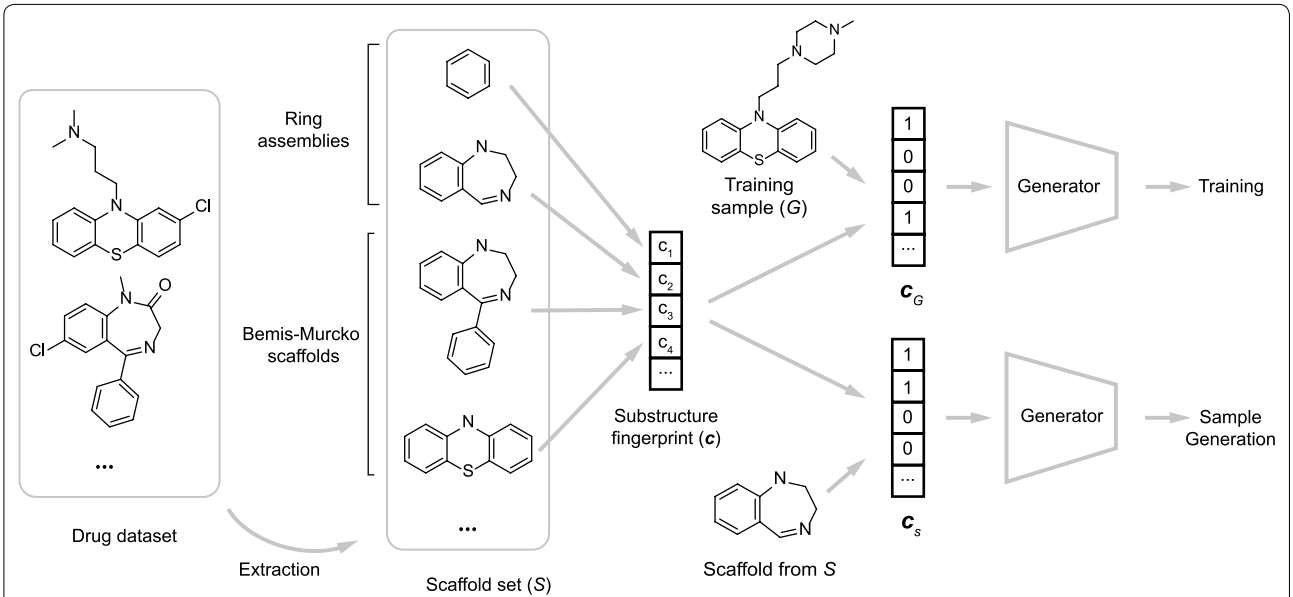

Fig. 7 Workflow for scaffold based molecule generation. Scaffold set S is first extracted from compounds in DrugBank. The conditional code $\mathbf{c}$ is set to be the substructure fingerprint based on S. Training is performed with the training samples labeled with $\mathbf{c}_G$. After training, scaffold based generation is performed using the fingerprint $\mathbf{c}_s$ of the query scaffold $s \in S$

is described as a set C instead of a single point $\mathbf{c}$. Here, we samples from $p_{\theta}(G|\mathbf{c} \in C)$ by first drawing $\mathbf{c}$ from $p(\mathbf{c}|\mathbf{c} \in C)$, and then drawing G from $p_{\theta}(G|\mathbf{c})$. Sampling from $p(\mathbf{c}|\mathbf{c} \in C)$ can be achieved by first sample $\mathbf{c}$ from $p(\mathbf{c})$ using molecules from the test set, then filter $\mathbf{c}$ according to the requirement $\mathbf{c} \in C$.

Designing dual inhibitor against JNK3 and GSK-3β

With the ability to model multiple requirements at once, conditional generative models can be used to design compounds with specific activity profiles for multiple targets. Here, we consider the task of designing dual inhibitors against both c-Jun N-terminal kinase 3 (JNK3) and glycogen synthase kinase-3 beta (GSK-3β). Both of the two targets are serine/threonine (S/T) kinases, and have shown to be related to the pathogenesis of various types of diseases [39, 40]. Notably, both JNK3 and GSK-3β are shown to be potential target in the treatment of Alzheimer's disease (AD). Jointly inhibiting JNK3 and GSK-3β may provide potential benefit for the treatment of AD.

The conditional code is set to be $\mathbf{c} = (c_{JNK3}, c_{GSK-3\beta})$, where c_{JNK3}, $c_{GSK-3\beta}$ are binary values indicating whether the compound is active against JNK3 and GSK-3β. For compounds in the ChEMBL dataset, c_{JNK3} and $c_{GSK-3\beta}$ are labeled using a separately trained predictor. Random forest (RF) classifier, which has been demonstrated to provide good performance for kinase activity prediction [41], is used as the predictor for GSK-3β and JNK3 activity. Here, we use ECFP6 [42] as the descriptor. The predictive model is trained using activity data from ExCAPE-DB [43], which is an integrated database with activity values from ChEMBL and PubChem [44]. Workflow for data extraction and predictor training is provided in Additional file 1: Supplementary Text 3. It is found that there is only 1.2% of molecules in ChEMBL that is predicted to be active against JNK3 or GSK-3β. This imbalance results in low enrichment rate during conditioned generation. For better result, the model is first trained under the unconditioned setting, and then fine-tuned based on the 1.2% molecules mentioned above.

Training details

The graph generative models are trained using the ChEMBL dataset. The data processing workflow largely follows Olivecrona et al. [13], as described in Additional file 1: Supplementary Text 1. MXNet [45] is used to implement the networks, and Adam optimizer [46] is used for network training. An initial learning rate of 0.001 is used together with a decay rate of 0.001 for every 100 iterations. Other parameters of the optimizer are set to be the default values suggested in [46] (that is, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$). The training lasts for 5 epochs, and the size of each mini-batch is set to 200 during the training.

During training, the decoding route is drawn from the distribution $q_{\alpha}(r|G)$. We tried three α values: 1.0, 0.8 and 0.6, as discussed previously. For $\alpha = 1.0$, k is set to 1 and the training can be performed on a single Nvidia GeForce GTX 1080Ti GPU for both MolMP and MolRNN. The training lasts for 14h for MolMP and 16h for MolRNN. For $\alpha = 0.8$ and $\alpha = 0.6$, k is set to 5 and the training is performed synchronously on 4 GPUs. The training lasts for 30h for MolMP and 35h for MolRNN.

For scaffold based and property based generation tasks, the conditonal graph generator is trained using the same setting as unconditional model. For the generation of GSK-3β and JNK3 inhibitors, the model is first trained using the full dataset, and the fine tuned on the subset that is predicted to be active against GSK-3β or JNK3. The fine-tuning uses a learning rate of 0.0001 and a decay rate of 0.002 for every 100 iterations. The fine-tuning lasts for 10 epochs, and takes 1h to finish.

In theory, the hyperparameters for the models mentioned above, including the training condition (batch size, learning rate, decay rate, β_1, β_2), model architectures(the number of convolutional layers, the hidden size in each layer) as well as α, should be optimized to achieve the best performance. However, due to the computational cost of both MolMP and MolRNN, we are unable to systematically optimize the hyperparameters. A througout discussion is only given for α, which determines the degree of randomness of q_{α}. No optimization is performed on model architecture except fitting it into the memory.

Comparison with SMILES based methods

The proposed graph-based model is compared with several SMILES based models. Two type of methods, variational autoencoder (VAE) and language model (LM), are considered in this comparison. The implementation of SMILES VAE follows Gómez-Bombarelli et al. [2]. The encoder contains three 1D convolutional layers, with 9, 9, 10 filters and 9, 9, 11 kernels each, and a fully connected layer with 435 hidden units. The model uses 196 latent variables and a decoder with three GRU layers with 488 hidden units. VAE for sequential data faces from the issue of "optimization challenge" [47, 48]. While the original implementation uses KL-annealing to tackle this problem, we follow the method provided by Kingma et al. [49] by controlling the level of free bits. This offers higher flexibility and stability compared with KL-annealing. We restrict the minimal level of free bits to 0.03 for each latent variable.

For LM, two types recurrent units are adopted. The first type uses GRU, and includes two architectures: the first architecture (SMILES GRU1) consists of three GRU layers with 512 hidden units each, and the second (SMILES GRU2), uses a wider GRU architecture with 1024 units, following the implementation by Olivecrona et al. [13]. Beside GRU, we also included a LSTM based SMILES language model following Segler et al. [12]. This architecture uses three LSTM layers, each with 1024 units.

Comparison with reinforcement learning (RL) based methods

We also compared the performance of conditional generative model with three RL based method. The first method, which is proposed by Olivecrona et al., maximizes following objective during model optimization:

$$G(\mathbf{x}) = -[\log p(\mathbf{x}) + \sigma S(\mathbf{x}) - \log q_\theta(\mathbf{x})]^2 \tag{18}$$

where $p(\mathbf{x})$ is the *Prior* network pre-trained using ChEMBL dataset, and $q(\mathbf{x})$ is the *Agent* network for task-specific molecule generation. SMILES GRU2 is used as the architecture for *Prior* and *Agent*.

This method is refered to as "REINVENT" [13]. We also include the following two baselines in the comparison. The first is a non-regularized RL method with the following objective:

$$G(\mathbf{x}) = \sigma S(\mathbf{x}) \tag{19}$$

We refer to this method as "Naive RL". The second method includes a prior term in addition to 19:

$$G(\mathbf{x}) = \sigma S(\mathbf{x}) + \log p(\mathbf{x}) \tag{20}$$

We refer to this method as "RL + Prior".

Evaluation metrics

Several metrics have been employed to evaluate the performance of generative models:

Sample validity

To test whether the generative models are capable of producing chemically correct outputs, 300,000 structures are generated for each model, and subsequently evalulated by RDKit for the rate of valid outputs. We also evaluate the ability of each model to produce novel structures. This is done by accessing the rate of generated compounds that do not occure inside the training set.

D_{KL} and D_{JS} for molecular properties

A good molecule generator should correctly model the distribution of important molecular properties. Therefore, the distribution of molecular weight (MW), log-partition coefficient (LogP) and QED between the generated dataset (p_g) and the test set (p_{data}) is compared for each method using Kullback–Leibler divergence (D_{KL}) and Jensen–Shannon divergence(D_{JS}):

$$D_{KL}(p_g||p_{data}) = \int_{\mathbb{R}} p_g(x) \log \frac{p_g(x)}{p_{data}(x)} dx \tag{21}$$

$$\begin{aligned} D_{JS}(p_g||p_{data}) = &\frac{1}{2} D_{KL}\left(p_g||\frac{p_g + p_{data}}{2}\right) \\ &+ \frac{1}{2} D_{KL}\left(p_{data}||\frac{p_g + p_{data}}{2}\right) \end{aligned} \tag{22}$$

D_{KL} and D_{JS} are widely used in deep generated models for both training [17, 50] and evaluation [51]. Here, the two values are determined using kernel density method implemented in SciPy [52]. We used a gaussian kernel with bandwidth selected based on Scott's Rule [53].

Performance metrics (R_c, $K_{cc'}$ and EOR_c) for task specific molecule design

For discrete conditional codes $\mathbf{c}$, let $M_\mathbf{c}$ be the set containing molecules sampled from distribution $p_\theta(G|\mathbf{c})$. $M_\mathbf{c}$ is obtained by first sampling molecule graphs conditioned on $\mathbf{c}$ and then removing invalid molecules. The size of $|M_\mathbf{c}|$ is set to 1000. Let $N_{\mathbf{cc}'}$ be the set of molecules in $M_\mathbf{c}$ that satisfy the condition $\mathbf{c}'$ ($\mathbf{c}'$ may be different from $\mathbf{c}$). The ratio $K_{\mathbf{cc}'}$ is defined as:

$$K_{\mathbf{cc}'} = \frac{|N_{\mathbf{cc}'}|}{|M_\mathbf{c}|} \tag{23}$$

The matrix $K_{\mathbf{cc}'}$ can be used to evaluate the ability of the model to control the output based on conditional code $\mathbf{c}$. When $\mathbf{c} = \mathbf{c}'$, this value gives the rate of correctly generated outputs, denoted by $R_\mathbf{c}$. High quality conditional models should have a high value of $R_\mathbf{c}$ and low values of $K_{\mathbf{cc}'}$ for $\mathbf{c} \neq \mathbf{c}'$. In paractice, we find that the value of $K_{\mathbf{cc}'}$ for scaffold and property based generation is significantly samller than $R_\mathbf{c}$ and have relatively low influence on the model's performance. Therefore, the result of $K_{\mathbf{cc}'}$ is omitted for scaffold and property based task, and is only reported for the task of kinase inhibitor design.

Let $R^0_\mathbf{c}$ be the rate of molecules in the training data that satisfy condition $\mathbf{c}$. The enrichment over random $EOR_\mathbf{c}$ is defined as:

$$EOR_\mathbf{c} = \frac{R_\mathbf{c}}{R^0_\mathbf{c}} \tag{24}$$

The definition is similar to that used in previous work [12], except that in their implementation $R^0_\mathbf{c}$ is calculated using the generated samples from the unconditioned model $p_\theta(G)$. For continuous codes, a subset C of the conditional code space is used to describe the generation

requirements. M_C is sampled from $p_\theta(G|\mathbf{c} \in C)$, and values for $K_{CC'}$, R_C and EOR_C can be calculated in a similar manner.

Rate of reproduced active compounds

For target based generation tasks, the rate of reproduced molecules is also reported following previous works [12, 13]. Take JNK3 as an example. During the evaluation, two sets of outputs are generated using two conditions: JNK3(+), GSK-3β (−) and JNK3(+), GSK-3β(+). The two set of outputs are denoted $M_{\mathbf{c}_1}$and $M_{\mathbf{c}_2}$respectively. Here, the size of $|M_{\mathbf{c}_1}|$ and $|M_{\mathbf{c}_2}|$ are both set to 50,000. Let T be the set containing the active molecules within the test set of JNK3. The rate of reproduced molecules (*reprod*) is calculated as:

$$reprod = \frac{|(M_{\mathbf{c}_1} \cup M_{\mathbf{c}_2}) \cap T|}{|T|} \tag{25}$$

For GSK-3β, the calculation can be done in a similar manner.

Sample diversity

For a good objective based molecule generator, the outputs are not only required to satisfy the given condition **c**, but also required to be structurally diverse. Benhenda [54] have suggested that the diversity of the model outputs should be consistent with the natural diversity of molecules satisfying the **c**. Also, Benhenda proposed to use the following statistics to evaluate the structural diversity of a given set of compounds:

$$I(M) = \frac{1}{|M|^2} \sum_{(x,y)\in M\times M} T_d(x,y) \tag{26}$$

where M is the set of sampled molecules, and $T_d(x,y)$ is the Tanimoto-distance between the two molecules x and y. $T_d(x,y)$ is defined using the Tanimoto-similarity T_s: $T_d(x,y) = 1 - T_s(x,y)$. This metric is called the internal diversity of the molecule set M.

For each condition **c**, the natural diversity $I^0_{\mathbf{c}}$ is first calculated using molecules in ChEMBL. The diversity of conditional outputs $I_{\mathbf{c}}$ is then calculated for each model. Note that when calculating $I^0_{\mathbf{c}}$ and $I_{\mathbf{c}}$, we only include molecules that satisfy the condition **c**. Finally, the value $|I_{\mathbf{c}} - I^0_{\mathbf{c}}|$ is compared among different models for their ability to reconstruct the natural compound diversity.

Results and discussion

Model performance and sample quality

Several randomly generated samples by MolRNN are grouped by molecular weight and shown in Fig. 8a–c. The qauntitative comparison between SMILES based and graph based models (MolMP and MolRNN) has been performed, and the results are summarized in Tables 1 and 2. We first analysed the model performance in terms of the rate of valid outputs and the rate of valid and novel outputs. It can be seen from the results that both MolRNN and MolMP outperform all SMILES based methods. It is also noted that changing α from 1.0 to 0.8 can significantly increase the rate of valid outputs for both MolMP and MolRNN. Further decreasing α produces only marginal effect. The high validity of output structures by graph-based model is not surprising as the generation of SMILES poses much stricter rules to the output compared with the generation of molecular graphs.

Figure 8d, e summarize respectively the common mistakes made by SMILES-based and graph-based model during generation. Results in Fig. 8d show that the most common cause of invalid output for SMILES based models is grammar mistakes, such as unclosed parentheses or unpaired ring numberings. But for the graph-based model, the majority of invalid output is caused by broken aromaticity, as demonstrated in Fig. 8f. This is likely a result of stepwise decoding pattern of graph-based models, as the decoder can only see part of the aromatic structure during generation, while the determination of aromaticity requires the information of the entire ring. It is also observed that mistakes related to atom valance are relatively minor, meaning that those rules are easy to learn using graph convolution.

Graph-based methods also have the advantage of giving highly interpretable outputs compared with SMILES. This means that a large portion of invalid outputs can be easily corrected if necessary. For example, broken aromaticity can be restored by literately refining the number explicit hydrogens of aromatic atoms, and unclosed aromatic rings can be corrected simply by connecting the two ends using a new aromatic bond. Though possible, those corrections may introduce additional bias to the output samples depending on the implementation, thus not adopted in the subsequent tasks.

Next, we investigate the ability for the generators to learn the distribution of molecular properties, as demonstrated in Table 2. Results have shown that MolRNN gives the best performance in D_{KL} and D_{JS} for molecular weight (MW) and QED, while SMILES GRU2 gives the best performance for LogP. For MolMP, although it is able to outperform SMILES GRU1 in the rate of valid outputs, it fails to give better performance in D_{KL} and D_{JS}. This observation suggest that the molecule level recurrent unit in MolRNN can significantly imporved the ability for the model to learn information about the data distribution.

When it comes to the influence of α to D_{KL} and D_{JS}, it is found that changing α from 1.0 to 0.8 can significantly

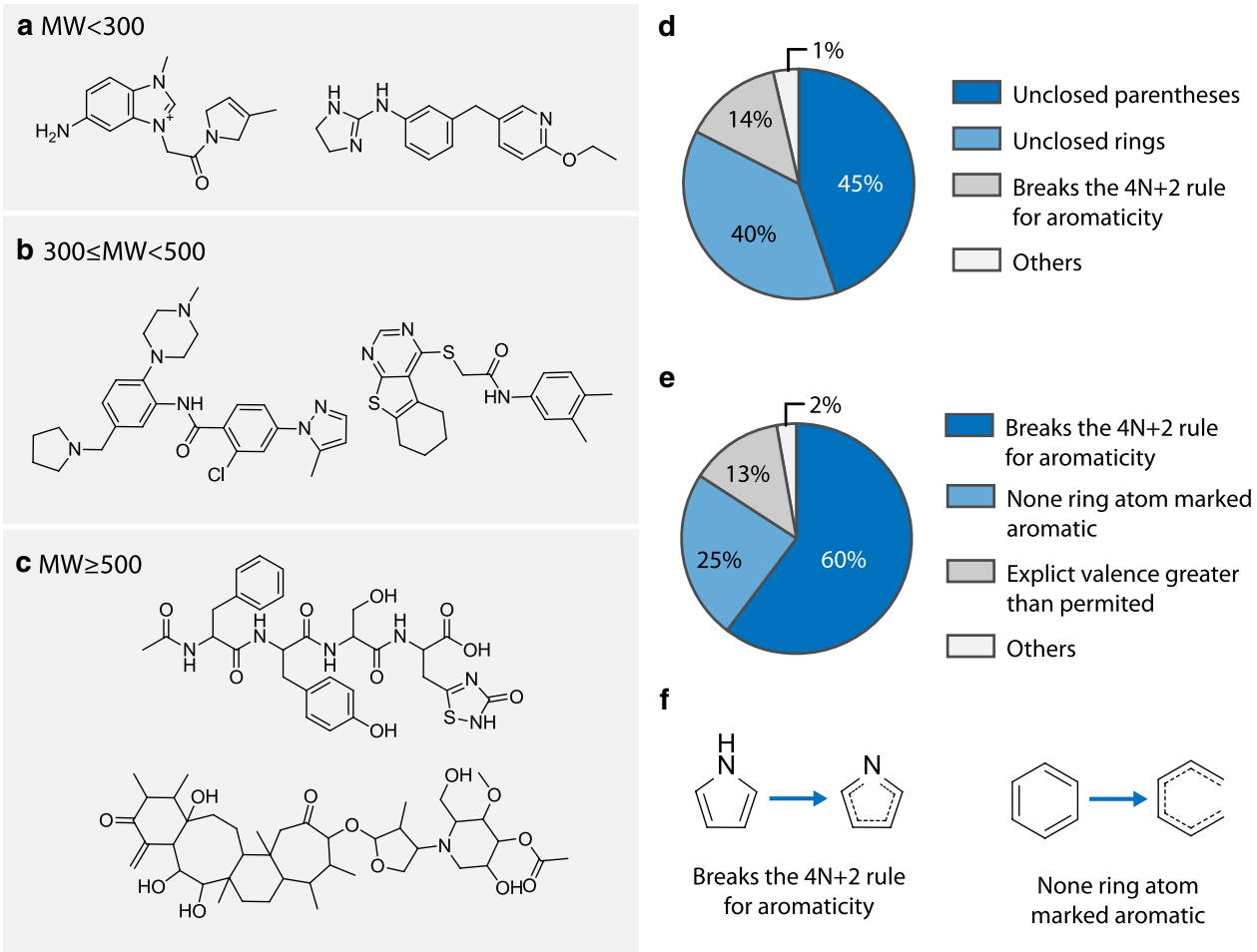

Fig. 8 A visualized demonstration of model outputs. **a**–**c**. Output samples by MolRNN. Results are grouped by molecular weight (**a** MW < 300, **b** 300 ≤ MW < 500, **c** MW ≥ 500); **d**, **e** Common mistakes made by SMILES based model and graph based model respectively; **f** Examples of broken aromaticity occurred during graph generation

Table 1 Comparison between SMILES based and graph-based generators in output validity

Model	% valid	% novel	% valid and novel
SMILES VAE	0.804 ± 0.016	0.986 ± 0.000	0.793 ± 0.016
SMILES GRU1	0.886 ± 0.002	0.984 ± 0.000	0.872 ± 0.002
SMILES GRU2	0.932 ± 0.002	0.965 ± 0.001	0.899 ± 0.002
SMILES LSTM	0.935 ± 0.006	0.975 ± 0.001	0.912 ± 0.006
MolMP ($\alpha = 1.0$)	0.952 ± 0.002	0.98 ± 0.001	0.933 ± 0.001
MolMP ($\alpha = 0.8$)	0.962 ± 0.002	0.984 ± 0.001	0.946 ± 0.001
MolMP ($\alpha = 0.6$)	0.963 ± 0.001	*0.988 ± 0.001***	0.951 ± 0.001
MolRNN ($\alpha = 1.0$)	0.967 ± 0.001	0.959 ± 0.000	0.928 ± 0.001
MolRNN ($\alpha = 0.8$)	*0.970 ± 0.001*	0.976 ± 0.001	0.947 ± 0.001
MolRNN ($\alpha = 0.6$)	*0.970 ± 0.001*	0.985 ± 0.000	*0.955 ± 0.001****

Results are reported as *Mean* ± *SD*. The best performance in each metric is highlighted in italics face. Also, for each metric, paired *t*-test is carried out for the difference between the best and second performing methods (*** for $p \leq 0.001$, ** for $p \leq 0.01$and * for $p \leq 0.05$). Multiple models are highlighted if the difference is not significant

improve the perforamnce of MolMP and MolRNN for all molecular properties. Further decreasing α to 0.6 will have different effect for MolMP and MolRNN. For MolMP, this will hurt the overall performance of D_{KL} and D_{JS}, while for MolRNN, this will inprove the performance for molecular weight, but will significantly decrease the performance of LogP. Overall, $\alpha = 0.8$ will be a better choise for MolMP, and $\alpha = 0.6$ will be more suited for MolRNN.

Generally, MolRNN have showed significant advantages among all generative mdoels considered. In the subsequent evaluation of conditonal generative models, the best performing graph based model (MolRNN) and the best performing SMILES based model (SMILES GRU2) are implemented as conditonal models and are compared among all tasks.

Table 2 Comparison between SMILES based and graph-based generators in D_{KL}(× 10^{-3}) and D_{JS}(× 10^{-3})

Model	MW		LogP		QED	
	D_{KL}	D_{JS}	D_{KL}	D_{JS}	D_{KL}	D_{JS}
SMILES VAE	13.5 ± 0.6	3.6 ± 0.2	3.9 ± 0.4	0.9 ± 0.1	2.6 ± 0.4	0.6 ± 0.1
SMILES GRU1	8.6 ± 0.4	2.3 ± 0.1	3.1 ± 0.3	0.7 ± 0.0	1.5 ± 0.3	0.3 ± 0.1
SMILES GRU2	7.8 ± 0:3	2:0 ± 0.1	*1.4 ± 0.2*	*0.3 ± 0.0*	2.2 ± 0.3	0.5 ± 0.1
SMILES LSTM	6.5 ± 0.7	1.8 ± 0.2	3.4 ± 1.2	0.8 ± 0.3	1.9 ± 1.3	0.4 ± 0.3
MolMP ($\alpha = 1.0$)	11.5 ± 1.3	3.4 ± 0.4	7.0 ± 1.8	1.7 ± 0.4	5.3 ± 1.2	1.3 ± 0.3
MolMP ($\alpha = 0.8$)	8.3 ± 1.6	2.4 ± 0.5	4.3 ± 1.2	0.9 ± 0.2	2.7 ± 0.8	0.6 ± 0.2
MolMP ($\alpha = 0.6$)	8.4 ± 1.0	2.4 ± 0.3	5.0 ± 1.3	1.1 ± 0.4	3.0 ± 0.9	0.7 ± 0.2
MolRNN ($\alpha = 1.0$)	5.0 ± 0.6	1.4 ± 0.2	2.8 ± 0.5	0.7 ± 0.1	2.0 ± 0.6	0.5 ± 0.1
MolRNN ($\alpha = 0.8$)	4.1 ± 0.7	1.1 ± 0.2	*1.6 ± 0.3*	*0.3 ± 0.1*	*1.0 ± 0.2*	*0.2 ± 0.0*
MolRNN ($\alpha = 0.6$)	*3.3 ± 0.2**	*0.9 ± 0.1***	3.0 ± 0.4	0.5 ± 0.1	*1.1 ± 0.4*	*0.2 ± 0.1*

Results are reported as *Mean* ± *SD*. The best performance in each metric is highlighted in italics face. Paired *t*-tests are carried out for the difference between the best and second performing methods (*** for $p \leq 0.001$, ** for $p \leq 0.01$ and * for $p \leq 0.05$). Multiple models are highlighted if the difference is not significant

Scaffold-based generation

In the first task, conditional generative models are trained to produce molecules based on given scaffolds. To illustrate the result, scaffold **1**, extracted from the antihypertensive drug Candesartan (see Fig. 9a), is used as an example, along with several related scaffolds (scaffold **2–4**) derived from scaffold **1** (Fig. 9a). Conditional codes **c** are constructed for each type of scaffold, and output structures are produced according to the corresponding code.

Results for both the SMILES based and graph based conditional generator are given in Table 3. In terms of output validity, graph based model is able to produce a higher fraction of valid outputs for scaffolds **1–4**, compared with SMILES based methods, which is similar to the results of unconditional models

In terms of the rate of correctly generated outputs ($R_{\mathbf{c}}$), although the models are unable to achieve 100% correctness, the $R_{\mathbf{c}}$ results are significantly higher than $R^0_{\mathbf{c}}$, offering high enrichment rate over random. Both graph based and SMILES based model are able to achieve $EOR_{\mathbf{c}} > 1000$ for scaffold **1–3** as well as $EOR_{\mathbf{c}} > 100$ for scaffold **4**, showing promising ability for the model to produce enriched output according to the given scaffold query. By comparing the result of $R_{\mathbf{c}}$ between the two type of architectures, it is found that graph based model have a higher performance for scaffold **3**, while SMILES based method have a higher performance for scaffold **2**. The two model have similar performance for scaffold **1** and scaffold **4**.

The structural diversity of the output samples is also evaluated for each model. Both graph based and SMILES based methods have resulted in a slightly lower output diversity $I_{\mathbf{c}}$ compared with the natural diversity $I^0_{\mathbf{c}}$. For scaffold **2**, the graph based method have better performance compared with SMILES based method, while for scaffold **4**, the SMILES based methods yields better result. For scaffold **1** and **3**, the difference is not significant between graph based and SMILES based method.

A comparison between conditional generative model and RL based approach is performed, using scaffold 4 as example. We set $\sigma = 20$, and formulate the score function $S_{\mathbf{c}}(\mathbf{x})$ as follows:

$$S_{\mathbf{c}}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} \text{ is valid and satisfies } \mathbf{c} \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

The result is summarized in Table 4. It is easily observed that all RL based approaches, including Naive RL, RL + Prior and REINVENT, are capable of achieving near perfect result on $R_{\mathbf{c}}$. However, in terms of output diversity, the RL based methods yields worse performance compared with conditional generative models. Among them, Naive RL result in the lowest output diversity of 0.468, followed by the RL + Prior, whose output diversity is 0.55. REINVENT results in a much higher output diversity of 0.750, but is still lower than that of conditional generative models.

The results above shows that conditional generators and RL based methods have opposite performance on $R_{\mathbf{c}}$ and $I_{\mathbf{c}}$. This is mainly caused by the fact that the two methods actually operate on different objectives. The former, which is trained under maximum likelihood estimation (MLE), optimizes $D_{KL}(p(\mathbf{x}|\mathbf{c})||q_\theta)$ (the proof is given in Additional file 1: Supplementary Text 5). During training, conditional generative model are encouraged to cover all modes in the data distribution, but are not punish for malicious modes, and therefore result in lower $R_{\mathbf{c}}$.

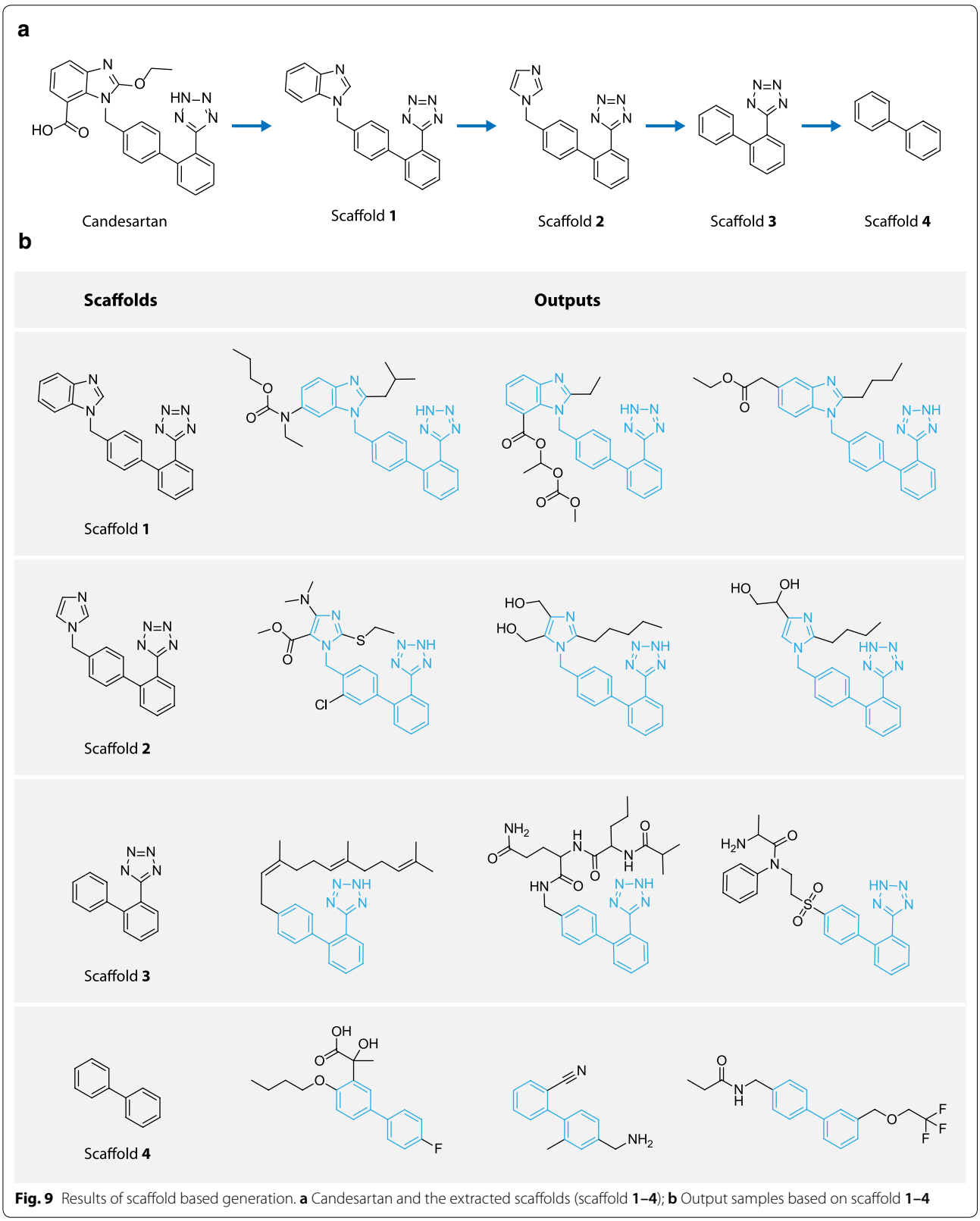

Fig. 9 Results of scaffold based generation. **a** Candesartan and the extracted scaffolds (scaffold **1**–**4**); **b** Output samples based on scaffold **1**–**4**

Table 3 Performance of graph based and SMILES based model on scaffold diversification tasks

Condition (c)	R_0	I_0	Model	% valid	R_c	EOR_c	Diversity (I_c)
scaffold **1**	7.9×10^{-5}	0.46	Graph	*0.931 ± 0.008*	0.86 ± 0.03	10865	0.45 ± 0.01
			SMILES	0.924 ± 0.005	*0.87 ± 0.01*	*10976*	*0.46 ± 0.01*
scaffold **2**	1.1×10^{-4}	0.50	Graph	*0.900 ± 0.016*	0.77 ± 0.04	6972	0.47 ± 0.02*
			SMILES	0.896 ± 0:011	*0.84 ± 0.01**	*7607*	*0.44 ± 0.01*
scaffold **3**	7.9×10^{-5}	0.56	Graph	*0.940 ± 0.019**	*0.56 ± 0.08***	*7086*	*0.60 ± 0.02*
			SMILES	0.898 ± 0.024	0.37 ± 0.07	4623	0.59 ± 0.03
scaffold **4**	5.8×10^{-3}	0.82	Graph	*0.982 ± 0.001****	0.88 ± 0.01	151	0.815 ± 0.001
			SMILES	0.969 ± 0.002	0.88 ± 0.00	151	*0.819 ± 0.00****

Results are reported as *Mean* ± *SD*. The best performance in each metric is highlighted in italics face. Paired *t*-tests are carried out for the difference between the graph and SMILES based method (*** for $p \leq 0.001$, ** for $p \leq 0.01$and * for $p \leq 0.05$)

Table 4 The comparison between condtional generative models RL based models in the task of generating molecules containing scaffold 4

Model	% valid	R_c	EOR_c	Diversity (I_c)
REINVENT	0.998 ± 0.000	*1.000 ± 0.000*	172	0.75 ± 0.01
Naive RL	0.984 ± 0.015	0.999 ± 0.001	172	0.48 ± 0.08
RL + Prior	*0.999 ± 0.001*	*1.000 ± 0.000*	172	0.55 ± 0.09
Graph	0.982 ± 0.001	0.88 ± 0.01	151	0.815 ± 0.001
SMILES	0.969 ± 0.002	0.88 ± 0.00	151	*0.819 ± 0.000*

Results are reported as *Mean* ± *SD*

The best performance in each metric is highlighted in italics face

The RL based approach, however, optimizes a completely different objective. It can be proved that maximizing Eq. 18 is equivalent to minimizing the reviersed KL divergence $D_{KL}(q_\theta||p(\mathbf{x}|\mathbf{c}))$. In fact, if the score $\sigma S(\mathbf{x})$ is formulated as $\log p(\mathbf{c}|\mathbf{x})$, which is the log-likelihood for the molecule $\mathbf{x}$ to satisfy the requirement $\mathbf{c}$, we can obtain the following relationship between $G(\mathbf{x})$ ans D_{KL}:

$$\nabla_\theta D_{KL}(q_\theta||p(\mathbf{x}|\mathbf{c})) = -\mathbb{E}_{\mathbf{x}\sim q_\theta}[\nabla_\theta G(\mathbf{x})] \tag{28}$$

The derivation is given in Additional file 1: Supplementary Text 6. This objective will force the model to comply with the given condition $\mathbf{c}$, but may result in potential mode collapse, and therefore lower output diversity. In short, conditonal generative model and RL based methods each emphasizes different aspect of the molecule distribution, and future research may explore the possibility to combine those methods.

Several generated samples by graph based model are given for each scaffold in Fig. 9b. Recall that the outputs given scaffold s should contain two type of molecules: (1) molecules with s as its Bemis–Murcko scaffold and (2) molecule whose Bemis–Murcko scaffold contains s but does not reside inside S. Both types are observed for scaffold **1**–**4** as shown in Fig. 9b. By further investigating the generated samples, it is observed that the model seems to have learnt about the side chains characteristics each scaffold. For example, samples generated from scaffold **1**–**3** usually have their substitutions occur at restricted positions, and frequently contains a long aliphatic side chain. Interestingly, this actually reflects the structural activity relationship (SAR) for angiotensin II (Ang II) receptor antagonists [55]. In fact, scaffold **1**–**3** have long been treated as a privileged structure against Ang II receptors [28], and as a result, molecules with scaffold **1**–**3** are largely biased to those who matches the SAR rules for the target. When trained with the biased dataset, the model can memorize the underlying structural activity relationship as a byproduct of scaffold based learning. This characteristic is beneficial for the generation of libraries containing specified privileged structures.

Generation based on drug-likeness and synthetic accessibility

In this task, the generative model is used to produce molecules according to the requirement on drug-likeness and synthetic accessibility. The conditional code is specified as $\mathbf{c} = (QED, SA)$. In the first experiments, the models are required to generate molecules based on the following requirements expressed as subsets of conditional code space: $C_1 = (0.84, 1) \times (0, 1.9)$, $C_2 = (0, 0.27) \times (0, 2.5)$, $C_3 = (0.84, 1) \times (3.4, +\infty)$ and $C_4 = (0, 0.27) \times (4.8, +\infty)$.

The values are determined from the distribution of QED and SA in ChEMBL dataset (see Fig. 10a) using the 90 and 10% quantile. The conditions are illustrated in Fig. 10d. The four sets represent four classes of molecules respectively and the first class C_1, which contains structures with high drug-likeness and high synthetic accessibility, defines the set of compounds that are most important for drug design.

Quantitative evaluations of graph based and SMILES based models are demonstrated in Table 5. Again, under

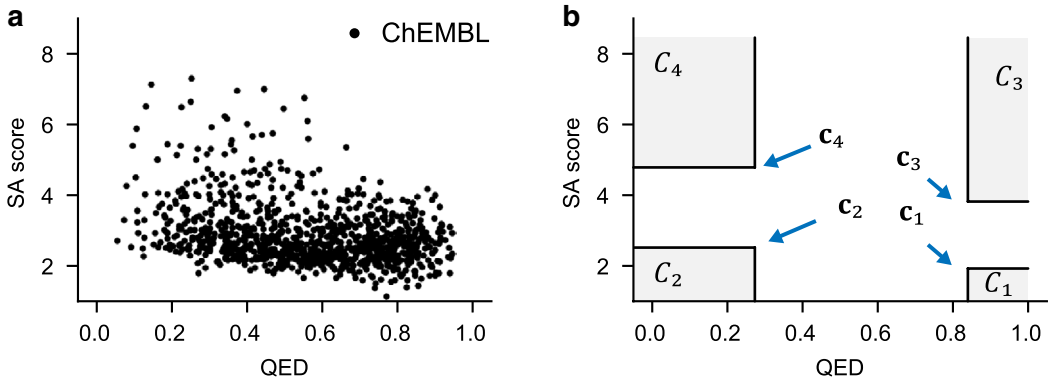

Fig. 10 Location of $C_1 \sim C_4$ and $c_1 \sim c_4$. **a** Distribution of QED and SAscore in the ChEMBL dataset; **b** Location of input conditions ($C_1 \sim C_4$ and $c_1 \sim c_4$)

all conditions ($C_1 \sim C_4$), the graph based model is able to outperform SMILES based model on the rate of valid outputs. The difference is most significant for conditions specifying low synthetic accessibility (that is, high SAscore, which is given by C_3 and C_4). This observation suggests that SMILES based model have difficulty in generating complex structures while maintaining the structural validity.

The graph based model also provides better performance in terms of R_C and EOR_C as shown in Table 5. It is noted that both graph and SMILES based models result in comparatively low R_C and EOR_C on condition C_3, which corresponds to molecules with high drug-likeness and low synthetic accessibility. However, this result is relatively easy to understand. Since the definition of drug-likeness contains the requriement for high synthetic accessibility, therefore finding molecules with high QED score and high SAscore is in itself a difficult task. For other conditions, the R_C results for both models varies from 50 to 70%. Those values are lower compared with scaffold based task, but nonetheless showing enrichments for all conditions over the distribution from ChEMBL. The diversity of generated samples are also reported.Different from the performance in %valid and $R_\mathbf{c}$, SMILES based method is able to produce outputs with slighly higher diversity compared with graph based method. The different is statistically significant for tasks C_1 and C_3.

We compared conditional generative model with RL based methods using C_1 as example. Similar to "Scaffold-Based Generation", we set σ to 20, and use the discrete score function $S_C(\mathbf{x})$ defined in Eq. 27. The results are summarized in Table 6. Overall, the performance of RL based methods are similar to that in the scaffold-based task. All RL methods are able to achieve high level of $R_\mathbf{c}$, but with lower output diversity.

For a visualized demonstration, the distributions of QED and SA score for the output samples from graph based generator are shown in Fig. 11. Random samples are also chosen for each class and are visualization in Fig. 12. The structural features for the output samples are mostly consistent with the predefined conditions, with small and simple molecules for C_1 and highly complex molecules for C_4.

Table 5 Performance of graph based and SMILES based model on property based generation tasks

Condition (C)	R_0	I_0	Model	% valid	R_C	EOR_C	Diversity (I_c)
C_1	0.009	0.810	Graph	*0.997 ± 0.000****	*0.55 ± 0.01****	*61*	0.798 ± 0.002
			SMILES	0.995 ± 0.001	0.51 ± 0.00	57	*0.806 ± 0.000****
C_2	0.012	0.850	Graph	*0.970 ± 0.002****	*0.55 ± 0.01***	*46*	0.841 ± 0.001
			SMILES	0.944 ± 0.001	0.52 ± 0.00	43	0.841 ± 0.001
C_3	0.011	0.868	Graph	*0.957 ± 0.001****	*0.35 ± 0.01***	*32*	0.864 ± 0.001
			SMILES	0.894 ± 0.007	0.31 ± 0.00	28	*0.866 ± 0.001***
C_4	0.008	0.867	Graph	*0.929 ± 0.003****	*0.73 ± 0.01***	*91*	0.863 ± 0.001
			SMILES	0.613 ± 0.015	0.66 ± 0.00	82	0.863 ± 0.000

Results are reported as *Mean* ± *SD*. The best performance in each metric is highlighted in italics face. Paired *t*-tests are carried out for the difference between the graph and SMILES based method (*** for $p \leq 0.001$, ** for $p \leq 0.01$and * for $p \leq 0.05$)

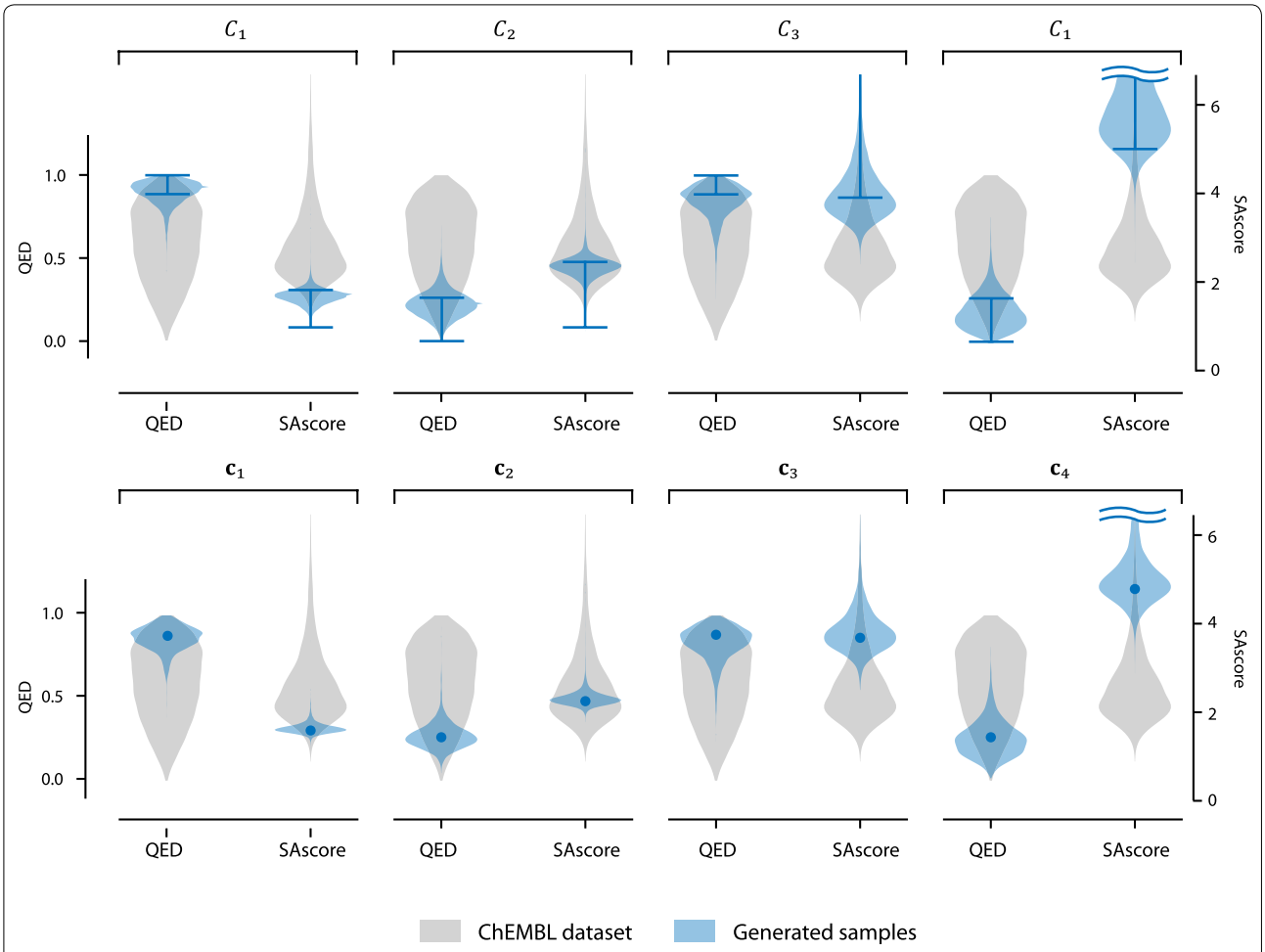

Fig. 11 Distribution of QED and SAscore for generated results: the upper row indicates distribution of QED and SAscore of molecules generated under conditions C_1, C_2, C_3 and C_4. The conditions $C_1 \sim C_4$ are shown as intervals represented by error bar. The lower row indicates distribution of QED and SAscore of molecules generated using single point conditions ($\mathbf{c_1}$, $\mathbf{c_2}$, $\mathbf{c_3}$ and $\mathbf{c_4}$). The conditions $\mathbf{c_1} \sim \mathbf{c_4}$ are represented as dots in the plot

Note that conditional models also support generation based on a given point of QED and SAscore. This is demonstrated visually using graph based conditional model. Now, the molecule generation process is conditioned on a single points of conditional code $\mathbf{c}$. Here, we use four different conditions as specified as follows: $\mathbf{c}_1 = (0.84, 1.9)$, $\mathbf{c}_2 = (0.27, 2.5)$, $\mathbf{c}_3 = (0.84, 3.8)$ and $\mathbf{c}_4 = (0.27, 4.8)$. Those conditons are also demonstrated in Fig. 10.

The distributions of QED and SAscore for the output molecules by graph based model are shown in Fig. 10e–h. Results show that, although the requirement is specified using a single value of QED and SAscore, the distribution of the two properties for output samples are relatively dispersed. This result is not surprising since the QED and SAscore score are relatively abstract descriptions of structural features of molecules, and a small modification of molecule structure may lead to significant changes in QED and SA scores. Nonetheless, it can be found that the generated samples are enriched around the corresponding code $\mathbf{c}$. It is also observed that the distribution of SAscore is more concentrated than that of QED. This is probably because that SAscore is direct measurement of molecular graph complexity, which may be easier to model for the graph based generator. In contrast, QED is a more abstract descriptor related to various molecular properties.

Generating dual inhibitors for JNK3 and GSK-3β

In this task, the models are used to generate dual inhibitor for JNK3 and GSK-3β. A predictive model is first used to label the conditional code for ChEMBL dataset, and the conditional graph generator is trained on the labeled training set. The two predictors yield good results in

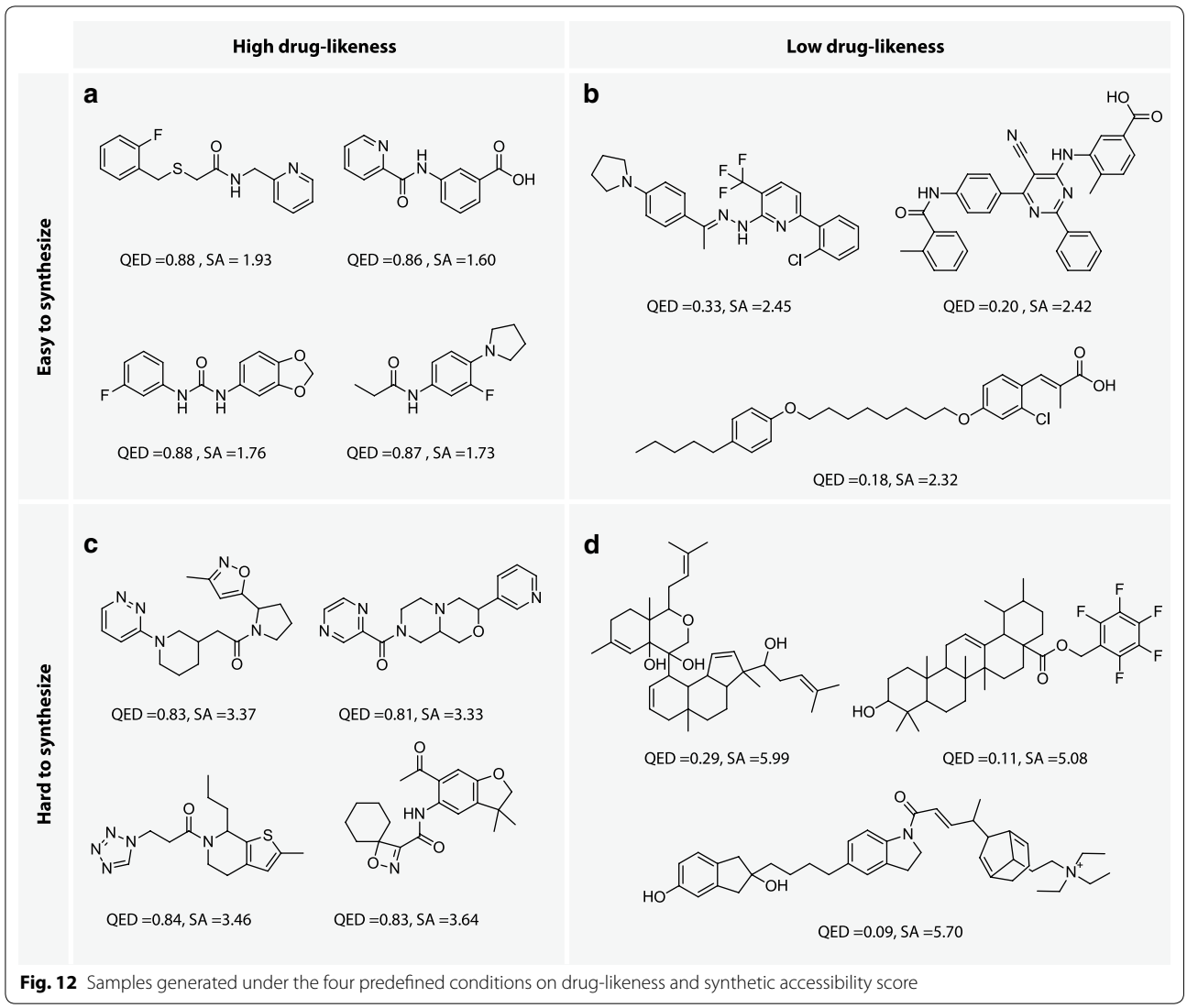

Fig. 12 Samples generated under the four predefined conditions on drug-likeness and synthetic accessibility score

Table 6 The comparison between condtional generative models RL based models in the task of generating molecules satisfying condition C_1 (that is, QED > 0.84 and SA score < 1.9)

Model	% valid	R_c	EOR_c	Diversity (I_c)
REINVENT	0.999 ± 0.001	0.986 ± 0.004	110	0.73 ± 0.07
Naive RL	0.993 ± 0.006	0.948 ± 0.052	105	0.64 ± 0.05
RL + Prior	*1.000 ± 0.000*	*0.999 ± 0.000*	*111*	0.44 ± 0.16
Graph	0.929 ± 0.003	0.73 ± 0.01	91	*0.863 ± 0.001*
SMILES	0.613 ± 0.015	0.66 ± 0.00	82	*0.863 ± 0.000*

Results are reported as *Mean* ± *SD*

The best performance in each metric is highlighted in italics face

general, with AUC = 0.983 for JNK3 and AUC = 0.984 for GSK-3β. The ROC curves for the two models are show in Additional file 2: Figure S4.

Results for both the SMILES based and graph based conditional generator are given in Table 7. In terms of output validity, graph based model outperforms SMILES based model in generating GSK-3β selective and JNK3 selective compounds, but for the generation of dual inhibitors, SMILES based model achieves better performance. In terms of R_c and EOR_c, SMILES based model is able to obtain better performance in generating dual inhibitors and selective inhibitors against GSK-3β, while the graph based model performs better in the task of generating JNK3 selective inhibitors.The $K_{cc'}$ matrices for graph based and SMILES based model are shown in Table 8. For both graph based and SMILES based model, it is noted that when generating compounds that is active to both JNK3 and GSK-3β, there is a significant amount of outputs falling into the category

Table 7 Performance of graph based and SMILES based model on inhibitor generation, results are reported as *Mean* ± *SD*

Condition (c)	R_0	l_0	Model	% valid	R_c	EOR_c	Diversity
GSK-3β(+)	0.0008	0.806	Graph	0.939 ± 0.007	0.53 ± 0.01	666	0.783 ± 0.006
JNK3(+)			SMILES	*0.959 ± 0.003***	*0.56 ± 0.01****	*697*	*0.784 ± 0.003*
GSK-3β(+)	0.01	0.860	Graph	*0.932 ± 0.007*	0.42 ± 0.01	42	0.851 ± 0.001
JNK3(−)			SMILES	0.928 ± 0.003*	*0.47 ± 0.01****	*47*	0.854 ± 0.001**
GSK-3β(−)	0.0008	0.829	Graph	*0.955 ± 0.003***	*0.61 ± 0.00****	*759*	0.814 ± 0.002
JNK3(+)			SMILES	0.944 ± 0.003	0.56 ± 0.01	698	*0.821 ± 0.001****

The best performance in each metric is highlighted in italics face. Paired *t*-tests are carried out for the difference between the graph and SMILES based method (*** for $p \leq 0.001$, ** for $p \leq 0.01$ and * for $p \leq 0.05$)

Table 8 The $K_{cc'}$ matrix for kinase inhibitor generation task, the diagnal elements $K_{cc} = R_c$ are omitted since they have been reported in Table 7

Condition (c)	Model	Results(c′)		
		GSK-3β(+), JNK3(+)	GSK-3β(+), JNK3(−)	GSK-3β(−), JNK3(+)
GSK-3β(+)	Graph	–	0.178 ± 0.007	*0.018 ± 0.001*
JNK3(+)	SMILES	–	*0.167 ± 0.010**	0.063 ± 0.006
GSK-3β(+)	Graph	*0.034 ± 0.001****	–	*0.003 ± 0.000****
JNK3(−)	SMILES	0.082 ± 0.007	–	0.023 ± 0.002
GSK-3β(−)	Graph	*0.024 ± 0.004****	*0.022 ± 0.002****	–
JNK3(+)	SMILES	0.083 ± 0.007	0.057 ± 0.002	–

Results are reported as *Mean* ± *SD*. The best performance in each metric is highlighted in italics face.Paired *t*-tests are carried out for the difference between the graph and SMILES based method (*** for $p \leq 0.001$, ** for $p \leq 0.01$ and * for $p \leq 0.05$)

of GSK-3β positive and JNK3 negative. Nonetheless, in terms of the enrichment over random EOR_c, the two models are able to achieve high performance for all selectivity combinations. Note that selective inhibitors for GSK-3β are relatively enriched in ChEMBL database, according to the result of the predictor. In comparison, the selective inhibitors against JNK3 and the dual inhibitor for both JNK3 and GSK-3β are much rarer. However, the model is still able to achieve significant enrichment for the two types of selectivity. The result shows potential application for target combinations that have low data enrichment rate.

Similar to previous tasks, a comparison with RL based methods is performed. Here, we mainly focus on the task to generate joint inhibitors to JNK3 and GSK-3β. In terms of the design of score function, we have employed $S_c(\mathbf{x})$ similar to that used in previous tasks (Eq. 27). The value of σ is set to 20 for S_d. The results are summarized in Table 9. Note that result for RL + Prior is omited, since in this task, we found that it tends to collapse quickly to a single molecule that could not provide meaningful result. The performance of Naive RL and REINVENT is similar to that reported in previous sections. Both RL based methods achieves high value of R_c, but have much lower output diversity.

Table 9 The comparison between condtional generative models RL based models in the task of generating dual inhibitors against GSK-3β and JNK3

Model	% valid	R_c	EOR_c	Diversity (l_c)
REINVENT	*0.999 ± 0.001*	*0.996 ± 0.005*	*1245*	0.3 ± 0.2
Naive RL	0.987 ± 0.007	0.969 ± 0.022	1211	0.4 ± 0.1
Graph	0.955 ± 0.003	0.61 ± 0.00	759	0.814 ± 0.002
SMILES	0.944 ± 0.003	0.56 ± 0.01	698	*0.821 ± 0.001*

Results are reported as *Mean* ± *SD*

The best performance in each metric is highlighted in italics face

To better demonstrate the structural distribution of the generated samples, visualization based on t-SNE [56] is performed using the ECFP6 fingerprint. The generated samples under different selectivity specifications and molecules in the test set for each target are projected into two-dimensional embeddings and are shown in Fig. 13a–d. According to the result, it is shown that the conditional generator tends to produce molecules near the test set samples, which is consistent with observations based on other methods [12]. It is also observed that molecules generated under different selectivity condition occupy distinct region of chemical space.

Fig. 13 Visualizing the distribution of generated samples for each target. The figure shows the t-SNE visualization of: **a** molecules form test set of GSK-3β and samples conditioned on JNK3(−), GSK-3β(+), **b** molecules from test set of GSK-3β and samples conditioned on JNK3(+), GSK-3β(+). **c** Molecules from test set of JNK3 and samples conditioned on JNK3(+), GSK-3β(−), **d** molecules from test set of JNK3 and samples conditioned on JNK3(+), GSK-3β(+)

For each selectivity condition, several molecules are sampled using the model and are demonstrated in Fig. 14a–c. By investigating the generated structures in detail, it can be observed that the model tends to generate samples containing well-established scaffold for the corresponding target. For JNK3, structures such as diaminopurines [57] and triazolones [58], which have been frequently used in the design of JNK inhibitors, show high occurrence in the generated samples. The observation is the same for GSK-3β, with example like 2,3-bis-arylmaleimides, a class of widely studied inhibitors of GSK-3 [59]. On the other hand, aminopyrimidines are frequently shown in the outputs of all selectivity conditions, but they are more enriched in generated dual inhibitors. Those observations show good interpretability of the outputs, and indicate that the structural features of generated samples are in line with the existing knowledge about the two targets.

Finally, we report the percentage of reproduced samples from the test set for each target. From the result, 10.3% of molecules are reproduced for JNK3 and, 6.0% of molecules are reproduced for GSK-3β. Note that molecules in the test sets for each targets have been excluded from the ChEMBL training set in this task, which means that the method is capable of generating molecules that have been confirmed to be positive, without seeing them in the training set of predictive model and conditional generative model.

Several recovered actives are shown in Fig. 14d–e. Those molecules show relatively high diversity in

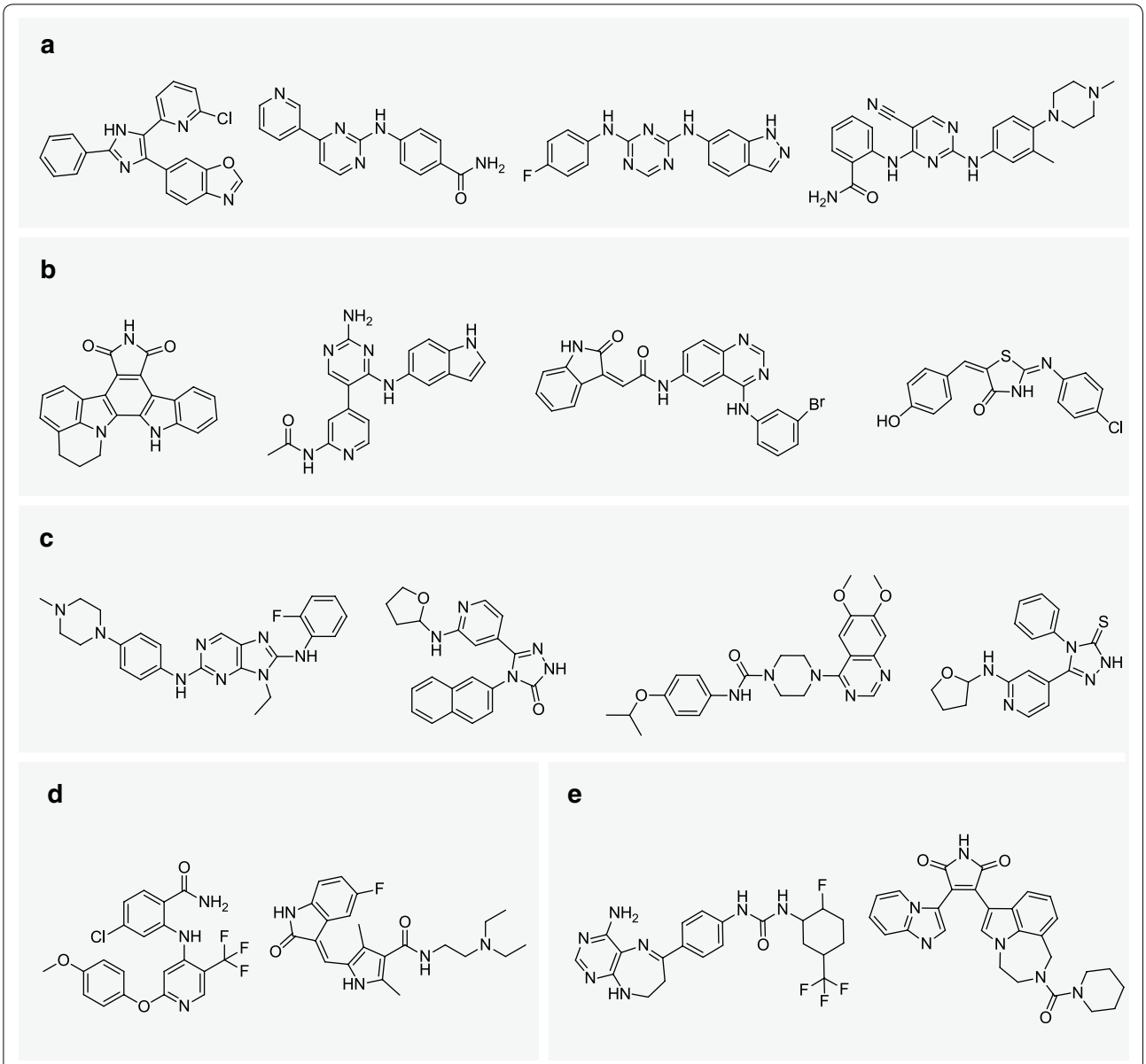

Fig. 14 Samples conditioned on different selectivity conditions. **a–c** Generated samples under different condition of selectivity (**a** for dual inhibitors, **b** for GSK-3β selective inhibitors, and **c** for JNK3 selective inhibitors); **d, e** Some recovered actives of JNK3 and GSK-3β respectively. **a** Generated dual inhibitors. **b** Generated GSK-3β selective inhibiors. **c** Generated JNK3 selective inhibiors. **d** Recovered JNK3 inhibitors. **e** Recovered GSK-3β inhibitors

structure, indicating that the model does not collapse to a subgroup of active compounds. A quantitative evaluation is performed using the internal diversity, and the result shows that the recovered GSK-3β inhibitors have a internal diversity of 0.819, while the recovered JNK3 inhibitors have a internal diversity of 0.761. Those values, although slighly lower, are relatively close to the diversity of test set molecules, which are 0.867 for GSK-3β and 0.852 for JNK3.

Conclusions

In this work, a new framework for de novo molecular design is proposed based on graph generative model and is applied to solve different drug design problems. The graph generator is designed to be more fitted to the tasks of molecule generation using a simple decoding scheme and a graph convolutional architecture that is less computationally expensive. Furthermore, a more flexible way of introducing decoding invariance is also suggested. The

method is trained using molecules in ChEMBL dataset and has demonstrated better performance compared with SMILES based methods, especially in terms of the rate of valid outputs.

To generate molecules with specific requirements, we propose to use conditional generative model, which offers high flexibility and do not require reinforcement learning. The model is applied to solve problems that is highly related to drug design, such as generating molecules based on a given scaffold, generating molecules with good drug-likeness and synthetic accessibility and the generation of molecules with specific profile against multiple targets. Results have showed that the conditional generative model can effectively produce enriched outputs based on the given requirements. A comparison with RL based method is performed, and results shows that although conditional generative model yields lower output accuracy, but it is capable of achieving higher output diversity.

This work can be extended in various aspects. First of all, the models used in this work completely ignores the stereochemistry information for molecules. In fact, stereochemistry is extremely important in the process of drug development, and introducing this information helps to improve the applicability of existing models. Secondly, for the target based generation, it will be much more helpful to jointly train the generator and the decoder, utilizing strategies such as semi-supervised learning [60, 61]. Finally, besides the three tasks experimented in this work, conditional graph generator can be used in many other scenarios. To summarize, the graph generative architecture proposed in this work gives promising result in various drug design tasks, and it is worthwhile to explore other potential applications using this method.

Additional files

Additional file 1. Containing additional information about the implementation details of experiments.

Additional file 2. Contianing supplementary figures.

Abbreviations

SMILES: simplified molecular-input line-entry system; RNN: recurrent neural network; LM: language model; RF: random forest; RL: reinforcement learning; VAE: variational autoencoder; GRU: gated recurrent unit; DRD2: dopamine receptor D2; JNK3: c-Jun N-terminal kinase 3; GSK3β: glycogen synthase kinase-3 beta; QED: quantitative estimate of drug-likeness; SA: synthetic accessibility; ECFP: extended connectivity fingerprint; t-SNE: t-distributed stochastic neighbor embedding; D_{KL}: Kullback–Leibler divergence; D_{JS}: Jensen–Shannon divergence.

Author's contributions

YL formulated the concept and contributed to the implementation. YL wrote the manuscript, LZ and ZL reviewed and edited the manuscript. All authors read and approved the final manuscript.

Acknowledgements

We would like to thank Xiaodong Dou for his help on the discussion of generated inhibitors of JNK3 and GSK3β. Thanks to Bo Yang who helped with the profiling of Additional file 1: Supplementary Text 8.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The materials supporing this article, including the source code, sampled molecules and pretrained models, are available at: https://github.com/kevinid/molecule_generator.

Funding

This research was supported by the National Natural Science Foundation of China (Grant Nos. 81573273, 81673279 21572010 and 21772005), the National Major Scientific and Technological Special Project for "Significant New Drugs Development" (2018ZX09735001-003) and Beijing Natural Science Foundation (7172118).

Ethics approval and consent to participate

Not applicable.

Consent for publication

The authors declare no competing fnancial interests.

Publisher's Note

Received: 22 February 2018 Accepted: 13 July 2018
Published online: 24 July 2018

References

1. Schneider G, Fechner U (2005) Computer-based de novo design of drug-like molecules. Nat Rev Drug Discov 4(8):649–663
2. Gómez-Bombarelli R, Duvenaud D, Hernández-Lobato JM, Aguilera-Iparraguirre J, Hirzel TD, Adams RP, Aspuru-Guzik A (2016) Automatic chemical design using a data-driven continuous representation of molecules. arXiv preprint arXiv:1610.02415v1
3. Böhm H-J (1992) The computer program ludi: a new method for the de novo design of enzyme inhibitors. J Comput Aided Mol Des 6(1):61–78
4. Mauser H, Stahl M (2007) Chemical fragment spaces for de novo design. J Chem Inf Model 47(2):318–324
5. Reutlinger M, Rodrigues T, Schneider P, Schneider G (2014) Multi-objective molecular de novo design by adaptive fragment prioritization. Angew Chem Int Ed 53(16):4244–4248
6. Hiss JA, Reutlinger M, Koch CP, Perna AM, Schneider P, Rodrigues T, Haller S, Folkers G, Weber L, Baleeiro RB (2014) Combinatorial chemistry by ant colony optimization. Future Med Chem 6(3):267–280
7. Dey F, Caflisch A (2008) Fragment-based de novo ligand design by multi-objective evolutionary optimization. J Chem Inf Model 48(3):679–690
8. Yuan Y, Pei J, Lai L (2011) Ligbuilder 2: a practical de novo drug design approach. J Chem Inf Model 51(5):1083–1091
9. Hartenfeller M, Proschak E, Schüller A, Schneider G (2008) Concept of combinatorial de novo design of drug-like molecules by particle swarm optimization. Chem Biol Drug Des 72(1):16–26
10. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Massachusetts
11. Lipton ZC, Berkowitz J, Elkan C (2015) A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019
12. Segler MH, Kogej T, Tyrchan C, Waller MP (2018) Generating focussed molecule libraries for drug discovery with recurrent neural networks. ACS Cent Sci 4(1):120–130
13. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. J Cheminform 9(1):48

14. Cho K, Van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. arXiv preprint arXiv:1406.1078
15. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, Light Y, McGlinchey S, Michalovich D, Al-Lazikani B (2011) Chembl: a large-scale bioactivity database for drug discovery. Nucleic Acids Res 40(D1):1100–1107
16. Popova M, Isayev O, Tropsha A (2017) Deep reinforcement learning for de-novo drug design. arXiv preprint arXiv:1711.10907
17. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114
18. And JJI, Shoichet BK (2005) Zinc: a free database of commercially available compounds for virtual screening. J Chem Inf Model 45(1):177
19. Blaschke T, Olivecrona M, Engkvist O, Bajorath J, Chen H (2018) Application of generative autoencoder in de novo molecular design. Mol Inform 37(1–2):1700123
20. Johnson DD (2017) Learning graphical state transitions. In: International conference on learning representations
21. Simonovsky M, Komodakis N (2018) Graphvae: towards generation of small graphs using variational autoencoders. arXiv preprint arXiv:1802.03480
22. Li Y, Vinyals O, Dyer C, Pascanu R, Battaglia P (2018) Learning deep generative models of graphs. In: International conference on learning representations
23. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. arXiv preprint arXiv:1603.05027
24. Wu Z, Ramsundar B, Feinberg EN, Gomes J, Geniesse C, Pappu AS, Leswing K, Pande V (2018) Moleculenet: a benchmark for molecular machine learning. arXiv preprint arXiv:1703.00564
25. Simonovsky M, Komodakis N (2017) Dynamic edge-conditioned filters in convolutional neural networks on graphs. arXiv preprint arXiv:1704.02901
26. Lima Guimaraes G, Sanchez-Lengeling B, Outeiral C, Cunha Farias PL, Aspuru-Guzik A (2017) Objective-reinforced generative adversarial networks (ORGAN) for sequence generation models. arXiv preprint arXiv:1705.10843
27. Neil D, Segler M, Guasch L, Ahmed M, Plumbley D, Sellwood M, Brown N (2018) Exploring deep recurrent models with reinforcement learning for molecule design. In: International conference on learning representations
28. Braese S (2015) Privileged scaffolds in medicinal chemistry:design, synthesis, evaluation. RSC Publishing, London
29. Bemis GW, Murcko MA (1996) The properties of known drugs. 1. Molecular frameworks. J Med Chem 39(15):2887–2893
30. Reis J, Gaspar A, Milhazes N, Borges FM (2017) Chromone as a privileged scaffold in drug discovery: recent advances. J Med Chem 60(19):7941–7957
31. Schuffenhauer A, Ertl P, Roggo S, Wetzel S, Koch MA, Waldmann H (2007) The scaffold tree: visualization of the scaffold universe by hierarchical scaffold classification. J Chem Inf Model 47(1):47–58
32. Varin T, Schuffenhauer A, Ertl P, Renner S (2011) Mining for bioactive scaffolds with scaffold networks: improved compound set enrichment from primary screening data. J Chem Inf Model 51(7):1528–1538
33. Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, Chang Z, Woolsey J (2006) Drugbank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Res 34(Database issue):668–672
34. Kadam R, Roy N (2007) Recent trends in drug-likeness prediction: a comprehensive review of in silico methods. Indian J Pharm Sci 69(5):609
35. Tian S, Wang J, Li Y, Li D, Xu L, Hou T (2015) The application of in silico drug-likeness predictions in pharmaceutical research. Adv Drug Deliv Rev 86:2–10
36. Ertl P, Schuffenhauer A (2009) Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. J Cheminform 1(1):8
37. Bickerton GR, Paolini GV, Besnard J, Muresan S, Hopkins AL (2012) Quantifying the chemical beauty of drugs. Nat Chem 4(2):90–98
38. RDKit: Open Source Cheminformatics. http://www.rdkit.org/
39. Koch P, Gehringer M, Laufer SA (2014) Inhibitors of c-jun N-terminal kinases: an update. J Med Chem 58(1):72–95
40. McCubrey JA, Davis NM, Abrams SL, Montalto G, Cervello M, Basecke J, Libra M, Nicoletti F, Cocco L, Martelli AM (2014) Diverse roles of gsk-3: tumor promoter-tumor suppressor, target in cancer therapy. Adv Biol Regul 54:176
41. Merget B, Turk S, Eid S, Rippmann F, Fulle S (2016) Profiling prediction of kinase inhibitors: toward the virtual assay. J Med Chem 60(1):474–485
42. Rogers D, Hahn M (2010) Extended-connectivity fingerprints. J Chem Inf Model 50(5):742–754
43. Sun J, Jeliazkova N, Chupakhin V, Golib-Dzib J-F, Engkvist O, Carlsson L, Wegner J, Ceulemans H, Georgiev I, Jeliazkov V (2017) Excape-db: an integrated large scale dataset facilitating big data analysis in chemogenomics. J Cheminform 9(1):17
44. Bolton EE, Wang Y, Thiessen PA, Bryant SH (2008) Pubchem: integrated platform of small molecules and biological activities. Annu Rep Comput Chem 4:217–241
45. Chen T, Li M, Li Y, Lin M, Wang N, Wang M, Xiao T, Xu B, Zhang C, Zhang Z (2015) Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. CoRR abs/1512.01274
46. Kingma D, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980
47. Bowman SR, Vilnis L, Vinyals O, Dai AM, Jozefowicz R, Bengio S (2015) Generating sentences from a continuous space. arXiv preprint arXiv:1511.06349
48. Chen X, Kingma DP, Salimans T, Duan Y, Dhariwal P, Schulman J, Sutskever I, Abbeel P (2016) Variational lossy autoencoder. arXiv preprint arXiv:1611.02731
49. Kingma DP, Salimans T, Jozefowicz R, Chen X, Sutskever I, Welling M Improved variational inference with inverse autoregressive flow. In: Advances in neural information processing systems, pp 4743–4751
50. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. arXiv preprint arXiv:1406.2661
51. Im DJ, Ma AH, Taylor GW, Branson K (2018) Quantitatively evaluating GANs with divergences proposed for training. In: International conference on learning representations
52. Jones E, Oliphant T, Peterson P et al (2001) SciPy: open source scientific tools for Python. http://www.scipy.org/
53. Scott DW (2008) Multivariate density estimation: theory, practice, and visualization. Wiley, New York
54. Benhenda M (2017) ChemGAN challenge for drug discovery: can AI reproduce natural chemical diversity? arXiv preprint arXiv:1708.08227
55. Almansa C, Gómez LA, Cavalcanti FL, de Arriba AF, García-Rafanell J, Forn J (1997) Synthesis and structure: activity relationship of a new series of potent at1 selective angiotensin ii receptor antagonists: 5-(biphenyl-4-ylmethyl) pyrazoles. J Med Chem 40(4):547–558
56. van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9(Nov):2579–2605
57. Krenitsky VP, Nadolny L, Delgado M, Ayala L, Clareen SS, Hilgraf R, Albers R, Hegde S, D'Sidocky N, Sapienza J (2012) Discovery of cc-930, an orally active anti-fibrotic JNK inhibitor. Bioorg Med Chem Lett 22(3):1433–1438
58. Probst GD, Bowers S, Sealy JM, Truong AP, Hom RK, Galemmo RA, Konradi AW, Sham HL, Quincy DA, Pan H (2011) Highly selective c-jun N-terminal kinase (JNK) 2 and 3 inhibitors with in vitro CNS-like pharmacokinetic properties prevent neurodegeneration. Bioorg Med Chem Lett 21(1):315–319
59. Osolodkin DI, Palyulin VA, Zefirov NS (2013) Glycogen synthase kinase 3 as an anticancer drug target: novel experimental findings and trends in the design of inhibitors. Curr Pharm Des 19(4):665–679
60. Kingma DP, Mohamed S, Rezende DJ, Welling M Semi-supervised learning with deep generative models. In: Advances in neural information processing systems, pp 3581–3589
61. Siddharth N, Paige B, de Meent V, Desmaison A, Wood F, Goodman ND, Kohli P, Torr PH (2017) Learning disentangled representations with semi-supervised deep generative models. arXiv preprint arXiv:1706.00400