DrugEx v3: Scaffold-Constrained Drug Design with Graph Transformer-based Reinforcement Learning

Xuhan Liu[1], Kai Ye[2], Herman W. T. van Vlijmen[1,3], Adriaan P. IJzerman[1], Gerard J. P. van Westen[1, *]

[1]Drug Discovery and Safety, Leiden Academic Centre for Drug Research, Einsteinweg 55, Leiden, The Netherlands

[2]School of Electrics and Information Engineering, Xi'an Jiaotong University, 28 XianningW Rd, Xi'an, China

[3]Janssen Pharmaceutica NV, Turnhoutseweg 30, B-2340, Beerse, Belgium

[*]To whom correspondence should be addressed: Gerard J. P. van Westen, Drug Discovery and Safety, Leiden Academic Centre for Drug Research, Einsteinweg 55, Leiden, The Netherlands. Tel: +31-71-527-3511. Email: gerard@lacdr.leidenuniv.nl.

Email Address of other authors: (1) Xuhan Liu: x.liu@lacdr.leidenuniv.nl; (2) Kai Ye: kaiye@xjtu.edu.cn; (3) Herman W. T. van Vlijmen: hvvlijme@its.jnj.com; (4) Adriaan P. IJzerman: ijzerman@lacdr.leidenuniv.nl.

Abstract

Due to the large drug-like chemical space available to search for feasible drug-like molecules, rational drug design often starts from specific scaffolds to which side chains/substituents are added or modified. With the rapid growth of the application of deep learning in drug discovery, a variety of effective approaches have been developed for *de novo* drug design. In previous work, we proposed a method named *DrugEx*, which can be applied in polypharmacology based on multi-objective deep reinforcement learning. However, the previous version is trained under fixed objectives similar to other known methods and does not allow users to input any prior information (*i.e.* a desired scaffold). In order to improve the general applicability, we updated *DrugEx* to design drug molecules based on scaffolds which consist of multiple fragments provided by users. In this work, the Transformer model was employed to generate molecular structures. The Transformer is a multi-head self-attention deep learning model containing an encoder to receive scaffolds as input and a decoder to generate molecules as output. In order to deal with the graph representation of molecules we proposed a novel positional encoding for each atom and bond based on an adjacency matrix to extend the architecture of the Transformer. Each molecule was generated by growing and connecting procedures for the fragments in the given scaffold that were unified into one model. Moreover, we trained this generator under a reinforcement learning framework to increase the number of desired ligands. As a proof of concept, our proposed method was applied to design ligands for the adenosine A_{2A} receptor ($A_{2A}AR$) and compared with SMILES-based methods. The results demonstrated the effectiveness of our method in that 100% of the generated molecules are valid and most of them had a high predicted affinity value towards $A_{2A}AR$ with given scaffolds.

Keywords: deep learning, reinforcement learning, policy gradient, drug design, Transformer, multi-objective optimization

Introduction

Due to the size of drug-like chemical space (*i.e.* estimated at 10^{33} - 10^{60} organic molecules) [1] it is impossible to screen every corner of it to discover optimal drug candidates. Commonly, the specific scaffolds derived from endogenous substances, high throughput screening, or a phenotypic assay [2] are taken as a starting point to design analogs while side chains/substituents are added or modified [3]. These fragments are used as "building blocks" to develop drug leads with e.g. combinatorial chemistry such as growing, linking, and merging [4]. After a promising drug lead has been discovered it is further optimized by modifying side chains to improve potency towards the relevant targets, selectivity over off-targets, and physicochemical properties which in turn can improve safety and tolerability [5].

In scaffold-based rational drug design, it is generally accepted that a chemical space consisting of 10^9 diverse molecules can be sampled with only 10^3 fragments [6] . For instance, one well known class of drug targets are G Protein-coupled receptors (GPCRS), a family via which approximately 35% of drug exert their effect [7]. The adenosine receptors (ARs) form a family within rhodopsin-like GPCRs and include four subtypes (A_1, A_{2A}, A_{2B} and A_3). Each of them has a unique pharmacological profile, tissue distribution, and effector coupling [8, 9]. ARs are ubiquitously distributed throughout the human tissues, and involved in many biological processes and diseases [10]. As adenosine is the endogenous agonist of ARs, a number of known ligands of the ARs are adenosine analogs and have a common scaffold. Examples include purines, xanthines, triazines, pyrimidines, and the inclusion of a ribose moiety [11]. In this work, we aim to design novel ligands for this family of receptors using a deep learning-based drug design method.

Deep learning based methods have been gaining ground in computational drug discovery, including *de novo* design, based on rapid developments over the last decade [12]. Deep learning has achieved breakthroughs in visual recognition, natural language processing, and other data-rich fields [13]. For distribution-directed issues, Gomez-Bombarelli *et al.* implemented variational autoencoders (VAE) to map molecules into a latent space where each point can also be decoded into unique molecules inversely [14]. They used recurrent neural networks (RNNs) to successfully learn SMILES (simplified molecular-input line-entry system) grammar and construct a distribution of molecular libraries [15]. For goal-directed issues, Sanchez-Lengeling *et al.* combined reinforcement learning and generative adversarial networks (GANs) to develop an approach named *ORGANIC* to design active compounds for a given target [16]. Olivecrona *et al.* proposed the *REINVENT* algorithm which updated the reinforcement learning with a Bayesian approach and combined RNNs to generate SMILES-based desired molecules [17, 18]. Moreover, Lim *et al.* proposed a method for scaffold-based molecular design with a graph generative model [19]. Li et al. also used deep learning to develop a tool named *DeepScaffold* for this issue [20]. Arús-Pous *et al.* employed RNNs to develop a SMILES-based scaffold decorator for *de novo* drug design [21]. Yang *et al.* used the Transformer model [22] to develop a tool named *SyntaLinker* for automatic fragment linking [23]. Here we continue to address on this issue further with different molecular representations and deep learning architectures.

In previous studies we investigated the performance of RNNs and proposed a method named *DrugEx* by integrating reinforcement learning to balance distribution-directed and goal-directed tasks [24]. Furthermore, we updated *DrugEx* with multi-objective reinforcement learning and applied it in polypharmacology [25]. However, the well-trained model cannot receive any input data from users and can only reflect the distribution of the desired molecules with fixed conditions. If the objectives are changed, the model needs to be trained again. In this work, we compared different end-to-end deep learning methods to update the *DrugEx* model to allow users to provide prior information, *e.g.* fragments that should occur in the generated molecules. Based on the

extensive experience in our group with the $A_{2A}AR$, we continue to take this target as an example to evaluate the performance of our proposed methods. In the following context, we will discuss the case of scaffold-constrained drug design, *i.e.* the model takes scaffolds composed of multiple fragments as input to generate desired molecules which are predicted to be active to $A_{2A}AR$. All python code for this study is freely available at http://gitlab.com/XuhanLiu/DrugEx.

Materials and Methods

Data source

The *ChEMBL* set was reused from our work on *DrugEx v2* [25]. This set consisted of small molecule compounds downloaded from ChEMBL using a SMILES notation (version 27) [26]. There were ~1.7 million molecules remained for model pre-training after data preprocessing implemented by RDKit. Preprocessing included neutralizing charges, removing metals and small fragments. In addition, 10,828 ligands and bioactivity data were extracted from ChEMBL to construct the *LIGAND* set, containing structures and activities from bioassays towards the four human adenosine receptors. The *LIGAND* set was used for fine-tuning the generative model. Molecules with annotated $A_{2A}AR$ activity were used to train a bioactivity prediction model. If multiple measurements for the same ligand existed, the average pChEMBL value (pX, including pKi, pKd, pIC50 or pEC50) was calculated and duplicate items were removed. In order to judge if the molecule is desired or not, the threshold of affinity was defined as pX = 6.5 to predict if the compound was active (>= 6.5) or inactive (< 6.5).

The dataset was constructed with an input-output pair for each data point. Each molecule was decomposed into a batch of fragments with the BRICS method [27] in RDKit (Fig 1A). If a molecule contained more than four leaf fragments, the smaller fragments were ignored and a maximum of four larger fragments were reserved to be randomly combined at one time. Their SMILES sequences were joined with '.' as input data which were paired with the full SMILES of molecules. Here, the scaffold was defined as the combination of different fragments which can be either continuous (linked) or discrete (separated). The resulting scaffold-molecule pairs formed the input and output data (Fig 1B). After completion of construction of the data pairs the set was split into a training set and test set with the ratio 9:1 based on the input scaffolds. The resulting *ChEMBL* set contained 10,418,681 and 1,083,271 pairs for training and test set, respectively. The *LIGAND* set contained 61,413 pairs in the training set and 7,525 pairs in the test set.

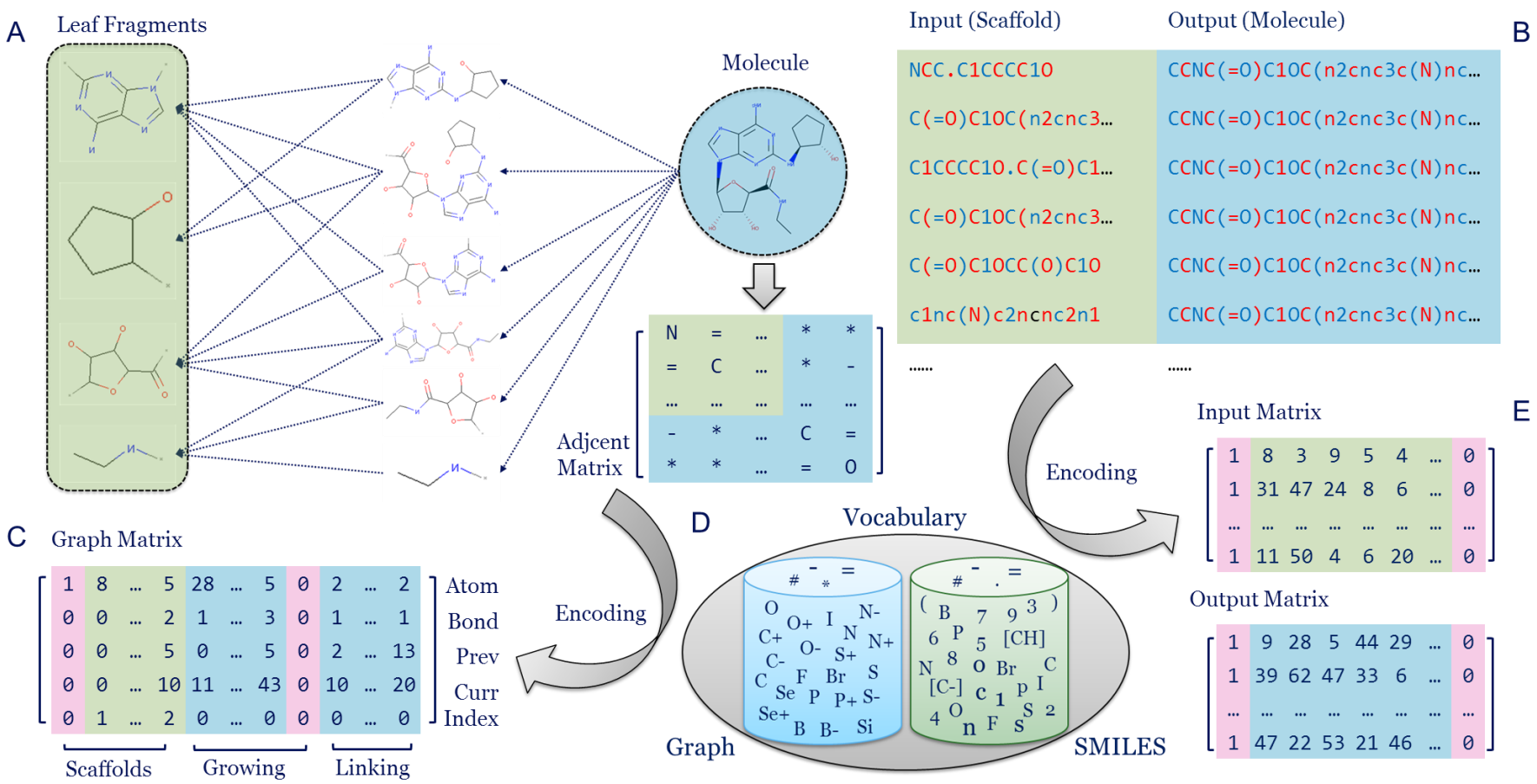

Fig. 1: scaffold-molecule pair dataset construction. (A) Each molecule in the dataset is decomposed hierarchically into a series of fragments with the BRICS algorithm. (B) Subsequently data pairs between input and output are created. Combinations of leaf fragments form the scaffold as input, while the whole molecule becomes the output. Each token in the SMILES sequences is separated by different colors. (C) After conversion to the adjacency matrix, each molecule was represented as a graph matrix. The graph matrix contains five rows, standing for the atom, bond, previous and current positions, and fragment index. The columns are composed with three parts to store the information of the scaffold, the growing section and the linking section. (D) All tokens are collected to construct the vocabularies for SMILES-based and graph-based generators, respectively. (E) An example of the input and output matrices for the SMILES representation of scaffolds and molecules

Molecular representations

In this study we tested two different molecular representations: SMILES and graph. For SMILES representations each scaffold-molecule pair was transformed into two SMILES sequences which were then split into different tokens to denote atoms, bonds, or other tokens for grammar control (e.g. parentheses or numbers). All of these tokens were put together to form a vocabulary which recorded the index of each token (Fig. 1D). Here, we used the same conversion procedure and vocabulary as in *DrugEx* v2 [25]. In addition, a start token (GO) was put at the beginning of a batch of data as input and an end token (END) at the end of the same batch of data as output. After sequence padding with a blank token at empty positions, each SMILES sequence was rewritten as a series of token indices with a fixed length. Subsequently all of these sequences for both scaffolds and molecules were concatenated to construct the input and output matrix (Fig. 1E).

For the graph representation each molecule was represented as a five-row matrix, in which the first two rows stand for the index of the atom and bond types, respectively. The third and fourth rows represent the position of previous and current atoms connected by a bond (Fig. 1C). The columns of this matrix contain three sections to store the scaffold, growing part, and linking part. The scaffold section began with a start token in the first row and the last row was labelled with the index of each scaffold starting from one. The scaffolds of each molecule are put in the beginning of the matrix, followed by the growing part for the scaffold, and the last part is the connecting bond between these growing fragments with single bonds. For the growing and linking sections the last row was always zero and these two sections were separated by the column of the end token. It is worth noticing that the last row was not directly involved in the training process. The vocabulary for graph representation was different from the SMILES representation, contains 38 atom types (Table S1), and four bond types (single, double, triple bonds and no bond). For each column, If an atom is the first occurrence in a given scaffold the type of the bond will be empty (indexed as 0 with token '*'). In addition, if the atom at the current position has occurred in the matrix, the type of the

atom in this column will be empty. In order to grasp more details of the graph representation, we also provided the pseudocode for encoding (Table S2) and decoding (Table S3).

End-to-End Deep learning

In this work, we compared three different sequential end-to-end DL architectures to deal with different molecular representations of either graph or SMILES (Fig. 2). These methods included: (A) a Graph Transformer, (B) an LSTM-based encoder-decoder model (LSTM-BASE), (C) an LSTM-based encoder-decoder model with an attention mechanism (LSTM+ATTN) and (D) a Sequential Transformer model. All of these DL models were constructed with *PyTorch* [28].

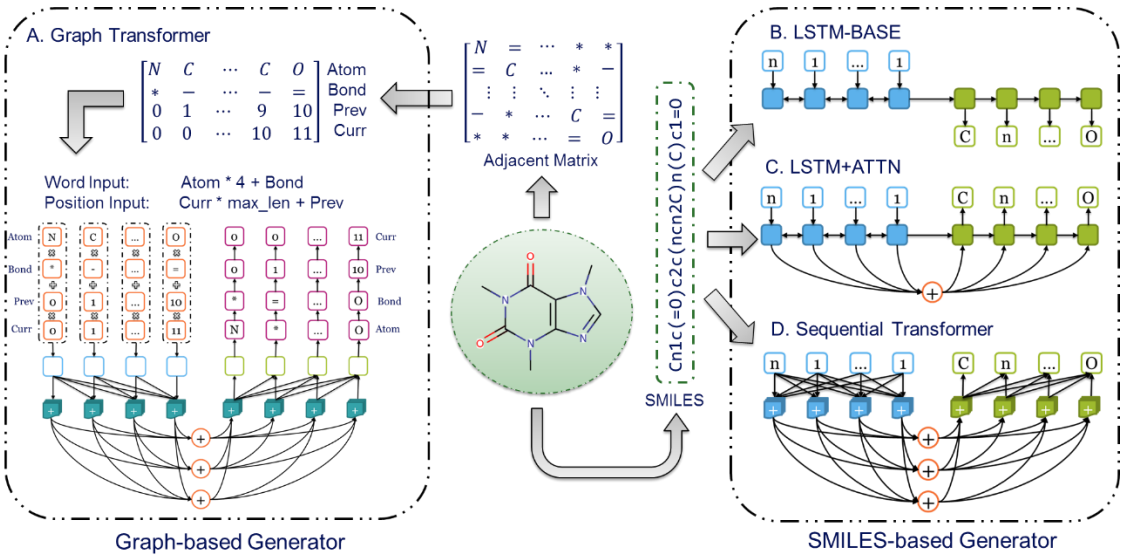

Fig. 2: Architectures of four different end-to-end deep learning models: (A) The Graph Transformer; (B) The LSTM-based encoder-decoder model (LSTM-BASE); (C) The LSTM-based encoder-decoder model with attention mechanisms (LSTM+ATTN); (D) The sequential Transformer model. The Graph Transformer accepts a graph representation as input and SMILES sequences are taken as input for the other three models.

For the SMILES representation based models three different types were constructed as follows (Fig. 2, right). The encoder and decoder in the LSTM-BASE model (Fig. 2B)

had the same architectures, containing one embedding layer, three recurrent layers, and one output layer (as used in *DrugEx v2*). The number of neurons in the embedding and hidden layers were 128 and 512, respectively. The hidden states of the recurrent layer in the encoder are directly sent to the decoder as the initial states. On the basis of the LSTM-BASE model an attention layer was added between the encoder and decoder to form the LSTM+ATTN model (Fig. 2C). The attention layer calculates the weight for each position of the input sequence to determine which position the decoder needs to focus on during the decoding process. For each step the weighted sums of the output calculated by the encoder are combined with the output of the embedding layer in the decoder to form the input for the recurrent layers. The output of the recurrent layers is dealt with by the output layer to generate the probability distribution of tokens in the vocabulary in both of these two models.

The sequential Transformer has a distinct architecture compared to the LSTM+ATTN model although it also exploits an attention mechanism. For the embedding layers "position encodings" are added into the typical embedding structure as the first layer of the encoder and decoder. This ensures that the model no longer needs to encode the input sequence token by token but can process all tokens in parallel. For the position embedding, sine and cosine functions are used to define its formula as follows:

$$PE_{(p,2i)} = \sin(pos/10000^{2i/d_m})$$

$$PE_{(p,2i+1)} = \cos(pos/10000^{2i/d_m})$$

where $PE(p, i)$ is the ith dimension of the position encoding at position p. It has the same dimension $d_m = 512$ as the typical embedding vectors so that the two can be summed.

In addition, self-attention is used in the hidden layers to cope with long-range dependencies. For each hidden layer in the encoder, it employs a residual connection around a multi-head self-attention sublayer and feed-forward sublayer followed by layer normalization. Besides these two sublayers in the decoder a third sublayer with multi-head attention is inserted to capture the information from output of the encoder.

This self-attention mechanism is defined as the scaled dot-product attention with three vectors: queries (Q), keys (K) and values (V), of which the dimensions are d_q, d_k, d_v, respectively. The output matrix is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^{\intercal}}{\sqrt{d_k}}\right)V$$

Instead of a single attention function, the Transformer adopts multi-head attention to combine information from different representations at different positions which is defined as:

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(head_1, \ldots, head_h)W^O$$

where h is the number of heads. For each head, the attention values were calculated by different and learned linear projections with Q, K and V as follows:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

where W^O, W^Q, W^K and W^V are metrics of learned weights and we set $h = 8$ as the number of heads and $d_k = d_v = 64$ in this work.

For the graph representation of the molecules we updated the sequential Transformer structure to propose a Graph Transformer (Fig. 2A). Similar to the sequential Transformer the Graph Transformer also requires the encodings of both word and position as the input. For the input word, the atom and bond cannot be processed simultaneously; therefore we combined the index of atom and bond together and defined it as follows:

$$I = I_{atom} \times 4 + I_{bond}$$

The index of the input word (I) for calculating word vectors is obtained by multiplying the atom index (I_{atom}) by four (the total number of bond types defined) and subsequently add the bond index (I_{bond}). Similarly, the position of each step cannot be used to calculate the position encoding directly. Faced with more complex data structure than sequential data, Dosovitskiy *et al.* proposed a new positional encoding scheme to define the position for each patch in image data for image recognition [29]. Inspired by their

work the position encoding at each step was defined as:

$$P = P_{curr} \times L_{max} + P_{prev}$$

The input position (P) for calculating the position encoding was obtained by multiplying the current position (P_{curr}) by the max length (L_{max}) and then adding the previous position (P_{prev}), which was then processed with the same positional encoding method as with the sequential Transformer. For the decoder, the hidden vector from the transformer was taken as the starting point to be decoded by a GRU-based recurrent layer; and the probability of atom, bond, previous and current position was decoded one by one sequentially.

When graph-based molecules are generated, the chemical valence rule is checked in every step. Invalid values of atom and bond types will be masked and an incorrect previous or current position will be removed ensuring the validity of all generated molecules. It is worth noticing that before being encoded, each molecule will be kekulized, meaning that the aromatic rings will be inferred to transform into either single or double bonds. The reason for this is that aromatic bonds interfere with the calculation of the valence value for each atom.

During the training process of SMILES-based models, a negative log likelihood function was used to construct the loss function to guarantee that the probability of the token at each step in the output sequence became large enough in the probability distribution of the vocabulary calculated by the deep learning model. In comparison, the loss function used by the Graph Transformer model also contains four parts for atom, bond, previous and current sites. Here the sum of these negative log probability values is minimized to optimize the parameters in the model. For this, the Adam algorithm was used for the optimization of the loss function. Here, the learning rate was set as 10^{-4}, the batch size was 256, and training steps were set to 20 epochs for pre-training and 1,000 epochs for fine-tuning.

Multi-objective optimization

In order to combine multiple objectives we exploited a Pareto-based ranking algorithm with GPU acceleration as mentioned in *DrugEx v2* [25]. Given two solutions m_1 and m_2 with their scores ($x_1, x_2, ..., x_n$) and ($y_1, y_2, ..., y_n$), then m_1 is said to Pareto dominate m_2 if and only if:

$$\forall \text{ j} \in \{1, \dots, \text{n}\}: x_j \geq y_j \; and \; \exists \text{ j} \in \{1, \dots, \text{n}\}: x_j > y_j$$

otherwise, m_1 and m_2 are non-dominated with each other. After the dominance between all pair of solutions being determined, the non-dominated scoring algorithm is exploited to obtain a rank of Pareto frontiers which consist of a set of solutions. After obtaining frontiers between dominant solutions, molecules were ranked based on the average Tanimoto-distance to other molecules instead of the commonly used crowding distance in the same frontier. Subsequently molecules with smaller average distances were ranked on the top. The final reward R^* is defined as:

$$R^* = \begin{cases} 0.5 + \dfrac{k - N_{undesired}}{2N_{desired}}, & if\ desired \\ \dfrac{k}{2N_{undesired}}, & if\ undesired \end{cases}$$

here k is the index of the solution in the Pareto rank. Rewards of undesired and desired solutions will be evenly distributed in (0, 0.5] and (0.5, 0.1], respectively.

In this work, we took two objectives into consideration: 1) the QED score [30] as implemented by RDKit (from 0 to 1) to evaluate the drug-likeness of each molecule (a larger value means more drug-like) ; 2) an affinity score towards the $A_{2A}AR$ which was implemented by a random forest regression model with Scikit-Learn [31] like in *DrugEx v2* [25]. The input descriptors consisted of 2048D ECFP6 fingerprints and 19D physico-chemical descriptors (PhysChem). PhysChem included: molecular weight, logP, number of H bond acceptors and donors, number of rotatable bonds, number of amide bonds, number of bridge head atoms, number of hetero atoms, number of spiro atoms, number of heavy atoms, the fraction of SP3 hybridized carbon atoms, number of aliphatic rings, number of saturated rings, number of total rings, number of aromatic rings, number of heterocycles, number of valence electrons, polar surface area, and

Wildman-Crippen MR value. Again it was determined if generated molecules are desired based on the Affinity score (larger than the threshold = 6.5). In addition, the SA score was also exploited an independent measurement to evaluate the synthesizability of generated molecules, which is also calculated by RDKit [32].

Reinforcement learning

In this work we constructed a reinforcement learning framework based on the interplay between a Graph Transformer (agent) and two scoring functions (environment). A policy gradient method was implemented to train the reinforcement learning model, the objective function is designated as follows:

$$J(\theta) = \mathbb{E}[R^*(y_{1:T})|\theta] = \sum_{t=1}^{T} logG(y_t|y_{1:t-1}) \cdot R^*(y_{1:T})$$

For each step t during the generation process the generator (G) determines the probability of each token (y_t) from the vocabulary to be chosen based on the generated sequence in previous steps ($y_{1:t-1}$). In the sequence-based models y_t can only be a token in the vocabulary to construct SMILES while it can be different type of atoms or bonds or the previous or current position in the graph-based model. The parameters in the objective function are updated by employing a policy gradient based on the expected end reward (R^*) received from the predictor. By maximizing this function the parameter θ in the generator can be optimized to ensure that the generator designs desired molecules which obtain a high reward score.

In order to improve the diversity and reliability of generated molecules, we implemented our exploration strategy for molecule generation during the training loops. In the training loop our generator is trained to produce the chemical space as defined by the target of interest. In this strategy there are two networks with the same architectures, an exploitation net (G_θ) and an exploration net (G_φ). G_φ did not need to be trained and its parameters are always fixed and it is based on the general drug-like chemical space for diverse targets obtained from ChEMBL. The parameters in G_θ on the other hand were updated for each epoch based on the policy gradient. Again an

exploring rate (ε) was defined with a range of [0.0, 1.0] to determine the percentage of scaffolds being randomly selected as input by G_φ to generate molecules. Conversely G_θ generated molecules with other input scaffolds. After the training process was finished G_φ was removed and only G_θ was left as the final model for molecule generation.

Performance evaluation

In order to evaluate the performance of the generators, four coefficients were calculated from the population of generated molecules (validity, accuracy, desirability, and uniqueness) which are defined as:

$$\text{Validity} = \frac{N_{valid}}{N_{total}}$$

$$\text{Accuracy} = \frac{N_{accurate}}{N_{total}}$$

$$\text{Desirability} = \frac{N_{desired}}{N_{total}}$$

$$\text{Uniqueness} = \frac{N_{unique}}{N_{total}}$$

here N_{total} is the total number of molecules, N_{valid} is the number of molecules parsed as valid SMILES sequences, $N_{accurate}$ is the number of molecules that contained all given scaffolds, $N_{desired}$ is the number of desired molecules that reach all required objectives, and N_{unique} is the number of molecules which are different from others in the dataset.

To measure molecular diversity, we adopted the Solow Polasky measurement as in the previous work. This approach was proposed by Solow and Polasky in 1994 to estimate the diversity of a biological population in an eco-system [33]. The formula to calculate diversity was redefined to normalize the range of values from [1, m] to (0, m] as follows:

$$I(A) = \frac{1}{|A|}\boldsymbol{e}^{\intercal}F(\boldsymbol{s})^{-1}\boldsymbol{e}$$

where A is a set of drug molecules with a size of $|A|$ equal to m, $\boldsymbol{e}$ is an m-vector of 1's and $F(\boldsymbol{s}) = [f(d_{ij}))]$ is a non-singular $m \times m$ distance matrix, in which $f(d_{ij})$ stands for the distance function of each pair of molecule provided as follows:

$$f(d) = e^{-\theta d_{ij}}$$

here we defined the distance d_{ij} of molecules s_i and s_j by using the Tanimoto-distance with ECFP6 fingerprints as follows:

$$d_{ij} = d(s_i, s_j) = 1 - \frac{|s_i \cap s_j|}{|s_i \cup s_j|},$$

where $|s_i \cap s_j|$ represents the number of common fingerprint bits, and $|s_i \cup s_j|$ is the number of union fingerprint bits.

Results and Discussion

Fragmentation of molecule

As stated we decomposed each molecule into a series of fragments with the BRICS algorithm to construct a fragment-molecule pair. Each organic compound can be split into retrosynthetically interesting chemical substructures with a compiled elaborate set of rules. For the *ChEMBL* and *LIGAND* sets, we respectively obtained 194,782 and 2,223 fragments. We further split the LIGAND set into three parts: active ligands ($LIGAND^{+}$, 2,638), inactive ligands ($LIGAND^{-}$, 2710) and undetermined ligands ($LIGAND^{0}$, 5480) based on the pX of bioactivity for $A_{2A}AR$. The number of fragments in these four datasets have a similar distribution (Fig. 3A) and there are approximately five fragments on average for each molecule with a 95% confidence between [0, 11] (Fig. 3A).

In the *LIGAND* set the three subsets have a similar molecular weight distribution of the fragments (Fig. 3B) with an average of 164.3 Da, smaller than in the *ChEMBL* set (247.3 Da). In order to check the similarity of these fragments we used the Tanimoto similarity calculation with ECFP4 fingerprints between each pair of fragments in the same dataset. We found that most of them were smaller than 0.5 indicating that they are dissimilar to each other (Fig. 3C). Especially, the fragments in the $LIGAND^{+}$ set have the largest diversity. Moreover, the distribution of different fragments in these three subsets of the *LIGAND* set are shown in Fig. 3D. The molecules in these three subsets have their unique fragments and share some common substructures.

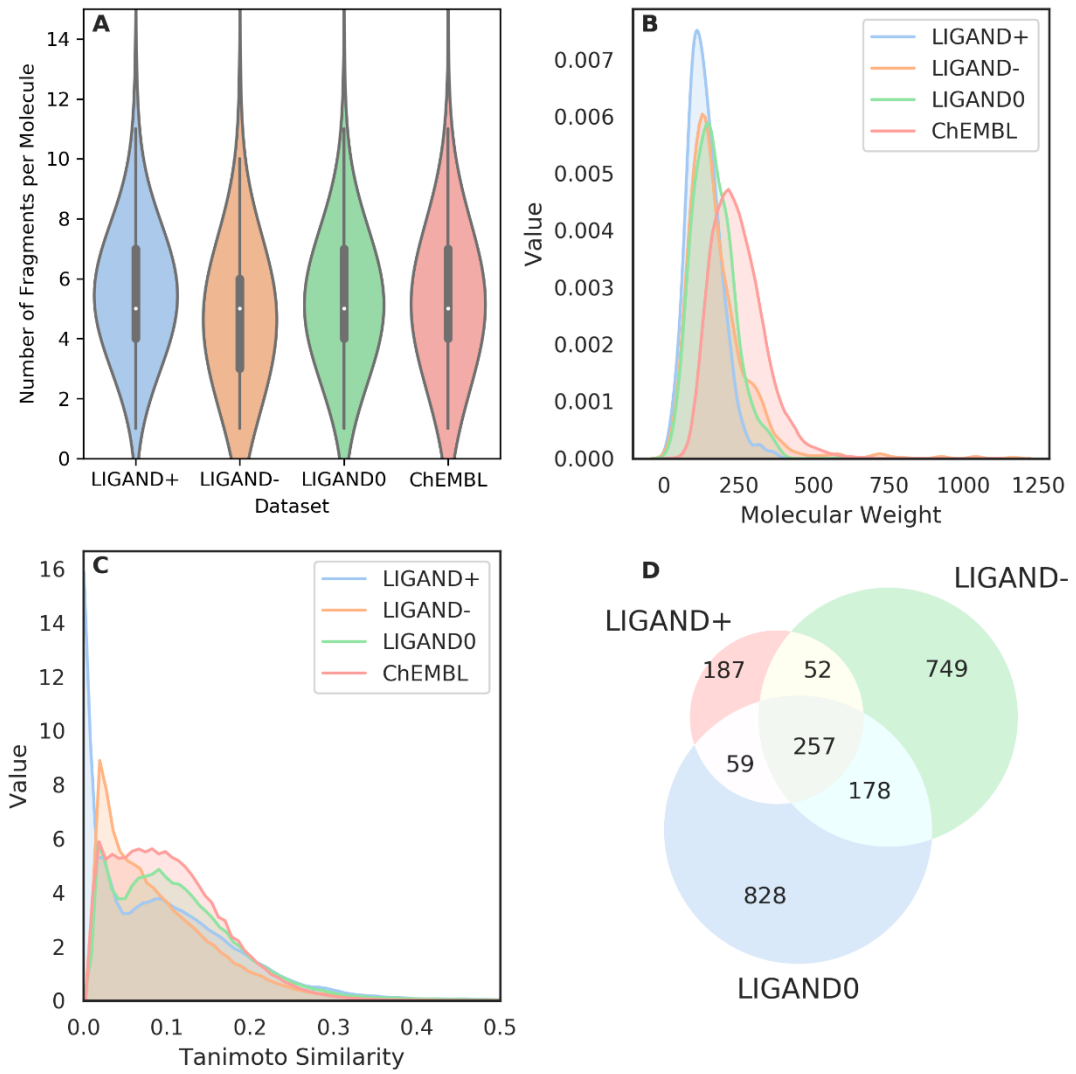

Fig 3: Analysis of some properties of fragments in the *ChEMBL* set and three *LIGAND* subsets. (A) Violin plot for the distribution of the number of fragments per molecules; (B) Distribution of molecular weight of these fragments; (C) Distribution of the similarity of the fragments measured by the Tanimoto-similarity with ECFP4 fingerprints; (D) Venn diagram for the intersection of the fragments existing in the three subsets of the *LIGAND* set.

Pre-training & Fine-tuning

After finishing the dataset construction four models were pre-trained on the *ChEMBL* set and fine-tuned on the *LIGAND* set. Here, these models were benchmarked on a server with four GTX1080Ti GPUs. After the training process converged, each fragment in the test set was presented as input for 10 times to generate molecules. The performance is shown in Table 1. The training of Transformer models was faster but consumed more computational resources than LSTM-based methods. In addition, Transformer methods outperformed LSTM-based methods using SMILES. Although the three SMILES-based models improved after being fine-tuned they were still outperformed by the Graph Transformer because of the advantages of a graph representation. To further check the accuracy of generated molecules we also compared the chemical space between the generated molecules and the compounds in the training set with three different representations 1) MW ~ logP; 2) PCA with 19D PhysChem descriptors; 3) tSNE with 2048D ECFP6 fingerprints (Fig. 4). The region occupied by molecules generated by the Graph Transformer overlapped completely with the compounds in both the *ChEMBL* and *LIGAND* sets.

Table 1: The performance of four different generators for pre-training and fine-tuning processes.

Methods	Pre-trained Model		Fine-tuned Model		Time	Memory
	Validity	Accuracy	Validity	Accuracy		
Graph Transformer	100%	99.3%	100%	99.2%	453.8 s	14.5 GB
Sequential Transformer	96.7%	72.0%	99.3%	87.3%	832.3 s	31.7 GB
LSTM-BASE	93.9%	44.1%	98.7%	77.9%	834.6 s	5.5 GB
LSTM+ATTN	89.7%	52.2%	96.4%	84.2%	1212.5 s	15.9 GB

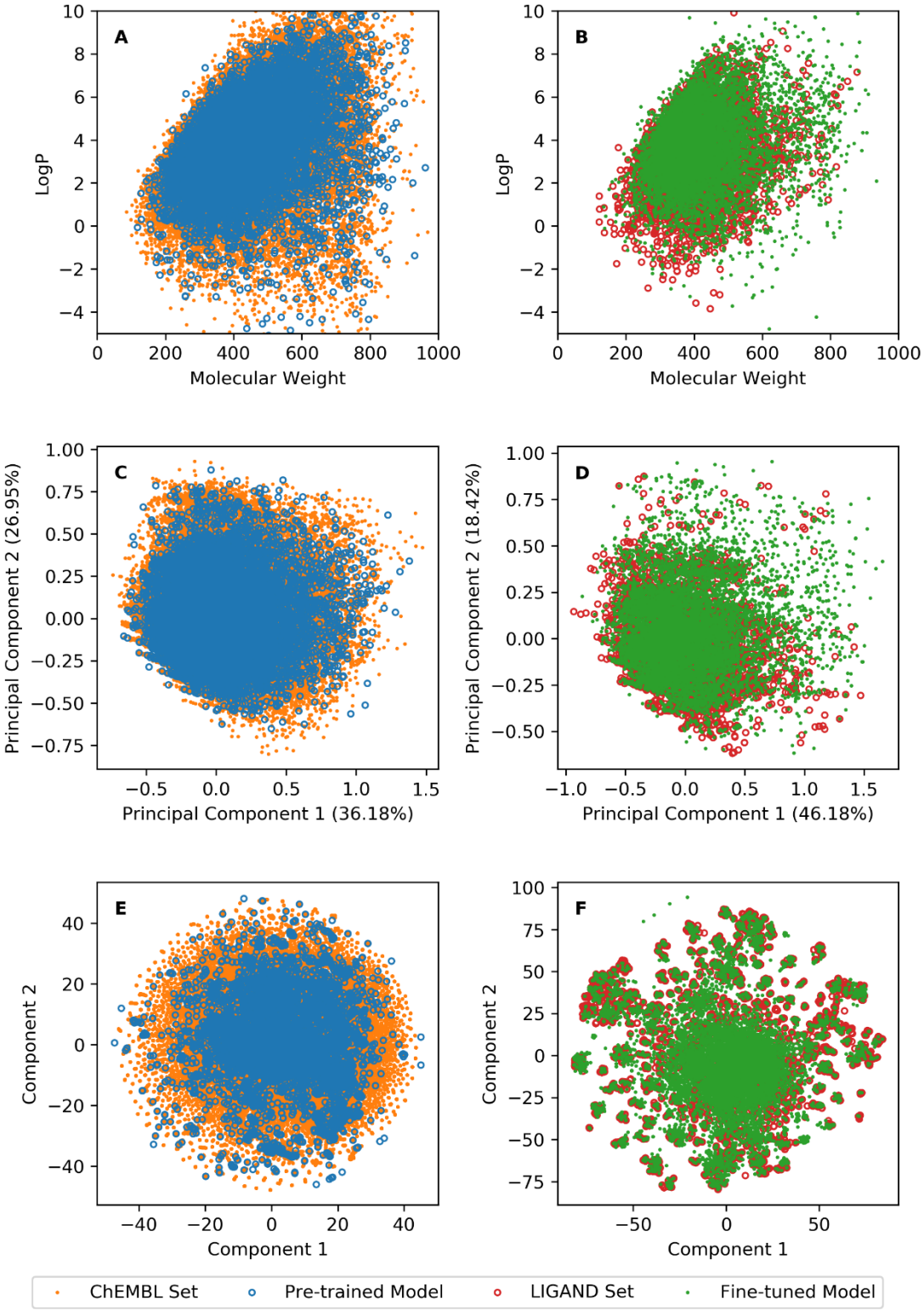

Fig. 4: The chemical space of generated molecules by the Graph Transformer pre-trained on the *ChEMBL* set (A, C and E) and being fine-tuned on the *LIGAND* set (B, D and F). Chemical space was represented by either logP ~ MW (A, B) and first two components in PCA on PhysChem descriptors (C, D) and t-SNE on ECFP6 fingerprints (E, F).

The graph representation for molecules has more advantages over the SMILES representation when dealing with fragment-based molecule design: 1) **Invariance in the local scale**: During the process of molecule generation, multiple fragments in a given scaffold can be put into any position in the output matrix without changing the order of atoms and bonds in that scaffold. 2) **Extendibility in the global scale**: When fragments in the scaffold are growing or being linked, they can be flexibly appended in the end column of the graph matrix while the original data structure does not need changing. 3) **Free of grammar**: Unlike in SMILES sequences there is no explicit grammar to constrain the generation of molecules, such as the parentheses for branches and the numbers for rings in SMILES; 4) **Accessibility of chemical rules**: For each added atom or bond the algorithm can detect if the valence of atoms is valid or not and mask invalid atoms or bonds in the vocabulary to guarantee the whole generated matrix can be successfully parsed into a molecule. With these advantages the Graph Transformer generates molecules faster while using less computational resources.

However, after examining the QED scores and SA scores we found that although the distribution of QED scores was similar between the methods (Figure 5A,C), the synthesizability of the molecules generated by the Graph Transformer were not better than the SMILES-based generators. This was especially true when fine-tuning on the *LIGAND* set. A possible reason is that molecules generated by the Graph Transformer contain uncommon rings when the model dealt with long-distance dependencies. In addition, because of more complicated data structure and presence of more parameters in the model, Graph Transformer did not outperform for the synthesizability of generated molecules when being trained on the small dataset (e.g. the *LIGAND* set). It is also worth noticing that there still was a small fraction of generated molecules that did not contain the required scaffolds which is caused by a kekulization problem. For example, a scaffold ‘CCC’ can be grown into ‘C1=C(C)C=CC=C1’. After being sanitized, it can be transformed into ‘c1c(C)cccc1’. In this process one single bond in the scaffold is changed to an aromatic bond, which causes a mismatch between the

scaffold and the molecule. Currently our algorithm cannot solve this problem because if the aromatic bond is taken into consideration, the valence of aromatic atoms is difficult to be calculated accurately. This would lead to the generation of invalid molecules. Therefore, there is no aromatic bond provided in the vocabulary and all of the aromatic rings are inferred automatically through the molecule sanitization method in RDKit.

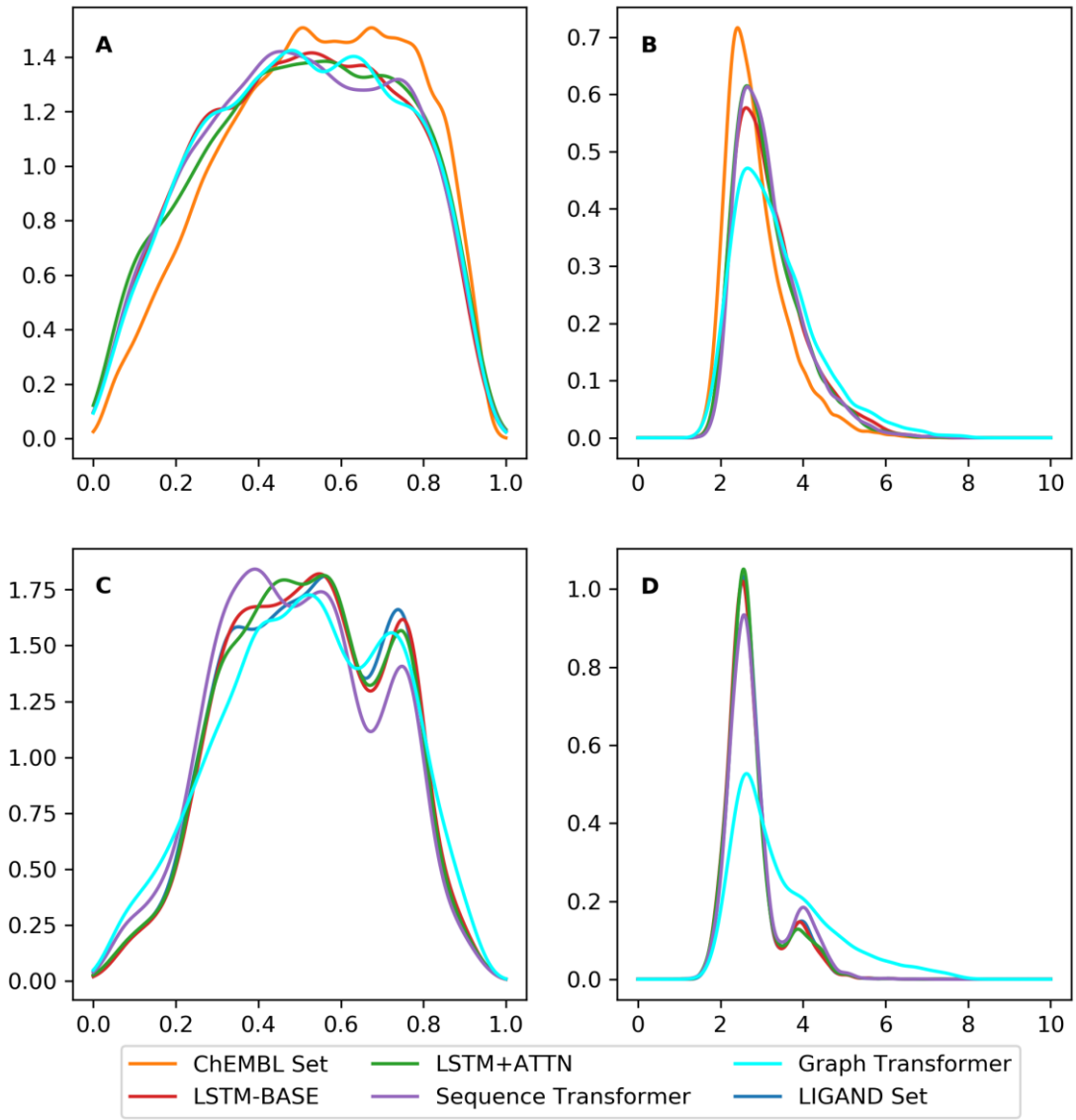

Fig. 5: the distribution of the QED score (A, C) and SA score (B, D) of desired ligands in the *ChEMBL* set and *LIGAND* set and of molecules generated by four different generators. For the QED score, four generators had the same performance as the molecules in both *ChEMBL* set (A) and the *LIGAND* set (C). For the SA score, Graph Transformer did not outperform three other SMILES-based generators in *ChEMBL* set (B) and even worse in the *LIGAND* set (D).

Policy gradient

Because the Graph Transformer generates molecules accurately and fast it was chosen as the agent in the RL framework. Two objectives were tested in the training process of this work. The first one was affinity towards $A_{2A}AR$, which is predicted by the random forest-based regression model from *DrugEx v2*; the second one was the QED score calculated with RDKit to measure how similar the generated molecule is to known approved drugs. With the policy gradient method as the reinforcement learning framework two cases were tested. On the one hand, predicted affinity for $A_{2A}AR$ was considered without the QED score. On the other hand, both objectives were used to optimize the model with Pareto ranking. In the first case 86.1% of the generated molecules were predicted active, while the percentage of predicted active molecules in the second case was 74.6%. Although the generator generated more active ligands without the QED score constraint most of them are not drug-like as they always have a molecular weight larger than 500Da. However, when we checked the chemical space represented by tSNE with ECFP6 fingerprints the overlap region between generated molecules and ligands in the training set was not complete implying that they fall out of the applicability domain of the regression model.

In *DrugEx v2*, we provided an exploration strategy which simulated the idea of evolutionary algorithms such as *crossover* and *mutation* manipulations. However, when coupled to the Graph Transformer there were some difficulties and we had to give up this strategy. Firstly, the mutation strategy did not improve with different mutation rates. A possible reason is that before being generated, part the molecule was fixed with a given scaffold counteracting the effect of mutation caused by the mutation net. Secondly, the *crossover* strategy is computationally very expensive in this context. This strategy needs the convergence of model training and iteratively updates the parameters in the agent. With multiple iterations, it takes a long period of time beyond the computational resources we can currently access. As a result, we updated the exploration strategy as mentioned in the Methods section with six different exploration

rates: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5].

Table 2: the performance of the Graph Transformer with different exploration rates in the RL framework.

ε	Accuracy	Desirability	Uniqueness	Diversity
0.0	99.7%	74.6%	60.7%	0.879
0.1	99.7%	66.8%	75.0%	0.842
0.2	99.8%	61.6%	80.2%	0.879
0.3	99.7%	56.8%	89.8%	0.874
0.4	99.7%	54.8%	88.8%	0.859
0.5	99.7%	46.8%	88.5%	0.875

Changes to the exploration rate do not influence accuracy and have a low effect on diversity. However, desirability (finding active ligands) and uniqueness can be influenced significantly. Empirically determining an optimal value for a given chemical space is recommended.

After training of the models, multiple scaffolds were input 10 times to generate molecules. The results for accuracy, desirability, uniqueness, and diversity with different exploration rates are shown in Table 2. With a low ε the model generates more desired molecules, but the uniqueness of the generated molecules can be improved. At ε = 0.3 the model generated the highest percentage of unique desired molecules (56.8%). Diversity was always larger than 0.84 and the model achieved the largest value (0.88) with ε = 0.0 or ε = 0.2. The chemical space represented by tSNE with ECFP6 fingerprints confirms that our exploration strategy produces a set of generated molecules completely covering the region occupied by the *LIGAND* set (Fig. 6).

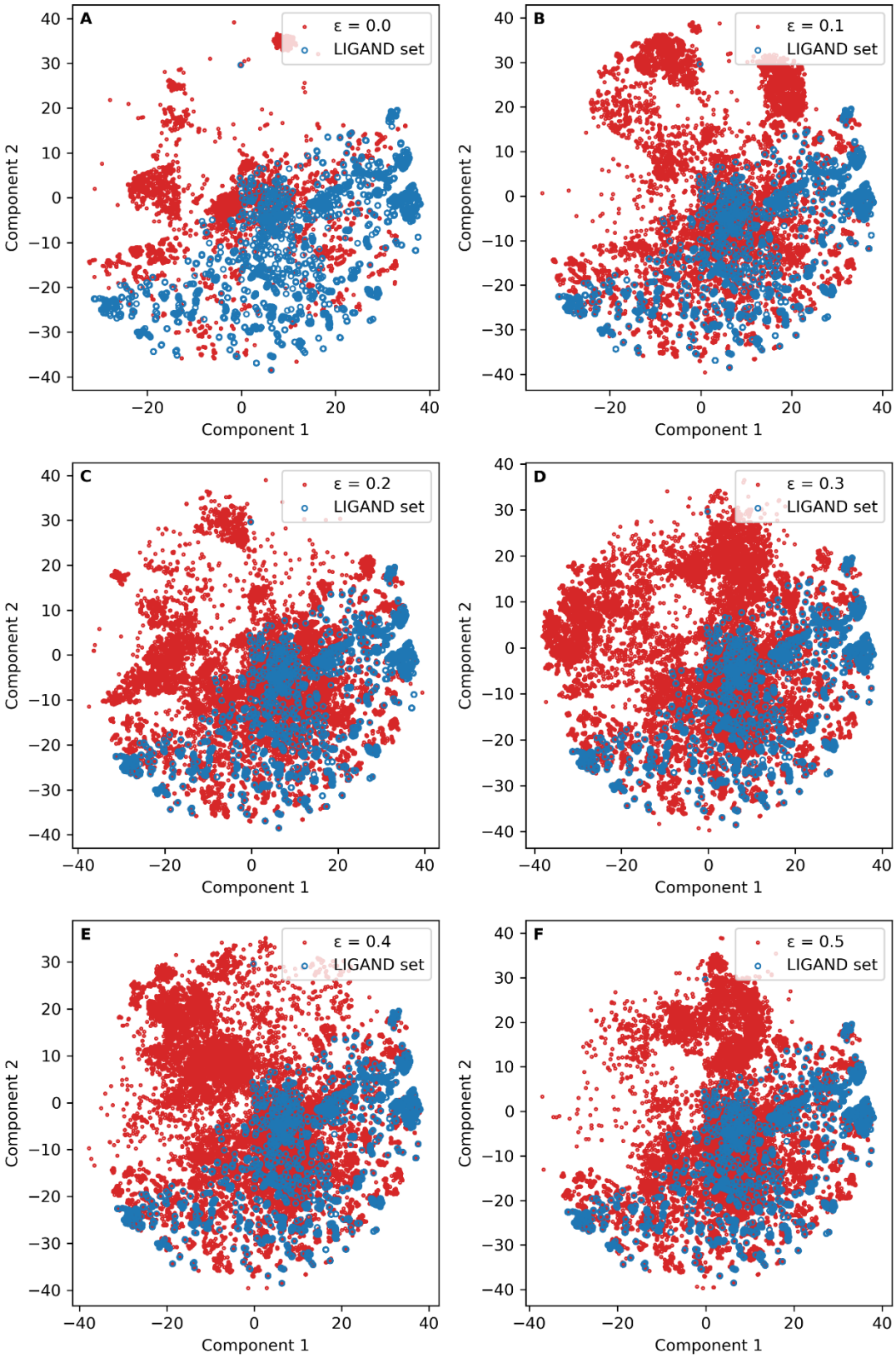

Fig. 6: The chemical space of generated molecules by the Graph Transformer trained with different exploration rates in the RL framework. The chemical space was represented by t-SNE on ECFP6 fingerprints.

Generated Molecules

In the chemical space making up antagonists of $A_{2A}AR$ there are several well-known scaffolds. Examples include furan, triazine, aminotriazole, and purine derivatives such as xanthine and azapurine. The Graph Transformer model produced active ligands for $A_{2A}AR$ (inferred from the predictors) with different combinations of these fragments as scaffolds. Taking these molecules generated by the Graph Transformer as an example, we filtered out the molecules with potentially reactive groups (such as aldehydes) and uncommon ring systems and listed 30 desired molecules as putative $A_{2A}AR$ ligands/antagonists (Fig. 7). For each scaffold five molecules were selected and assigned in the same row. These molecules are considered a valid starting point for further considerations and work (e.g. molecular docking or simulation).

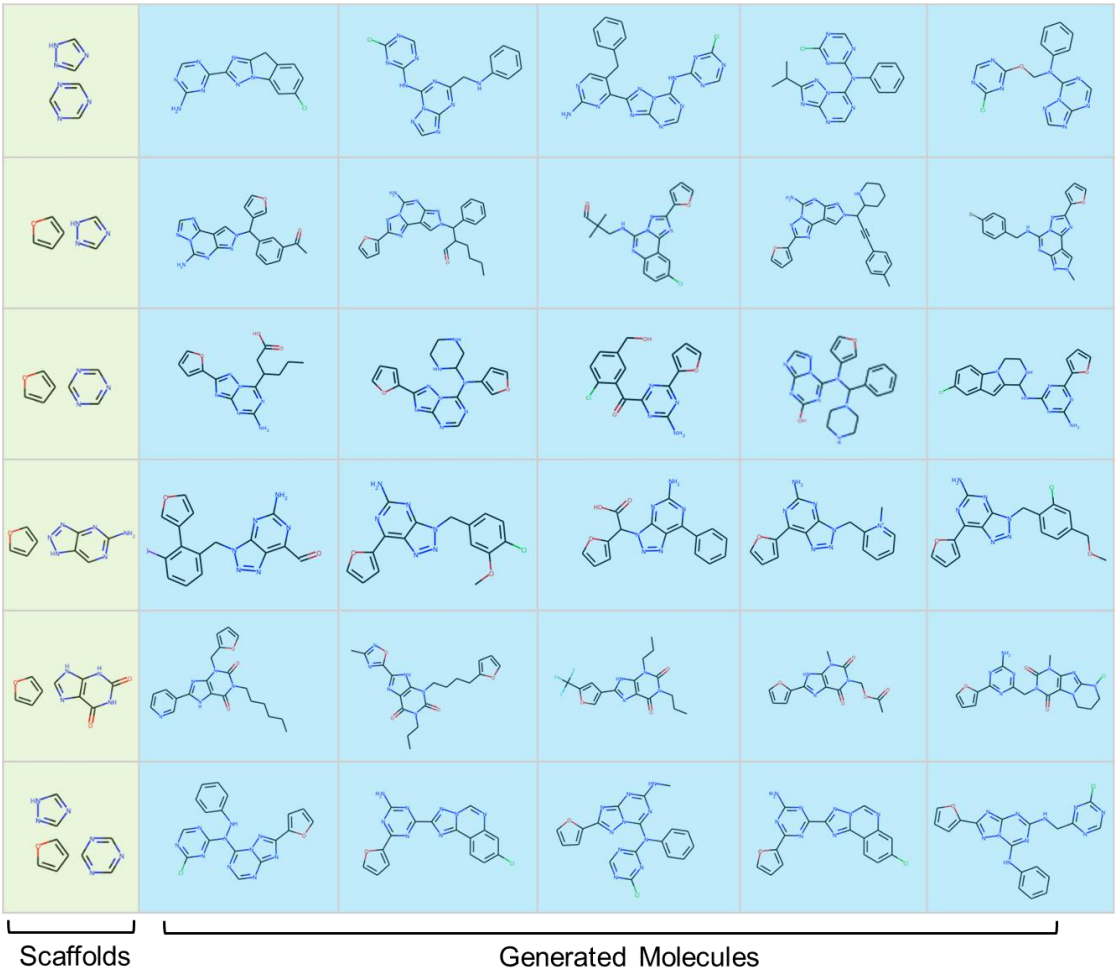

Fig. 7: Sample of generated molecules with the Graph Transformer with different scaffolds.

These scaffolds include: furan, triazine, aminotriazole, xanthine, and azapurine. The generated molecules based on the same scaffolds are aligned in the same row.

Conclusions and Future Perspectives

In this study, *DrugEx* was updated with the ability to design novel molecules based on scaffolds consisting of multiple fragments as input. In this version (*v3*), a new positional encoding scheme for atoms and bonds was proposed to make the Transformer model deal with a molecular graph representation. With one model, multiple fragments in a given scaffold can be grown at the same time and connected to generate a new molecule. In addition, chemical rules on valence are enforced at each step of the process of molecule generation to ensure that all generated molecules are valid. These advantages are impossible to be embodied in SMILES-based generation, as SMILES-based molecules are constrained by grammar that allows a 2D topology to be represented in a sequential way. With multi-objective reinforcement learning the model generates drug-like ligands, in our case for the $A_{2A}AR$ target.

In future work, the Graph Transformer will be extended to include other information as input to design drugs conditionally. For example, proteochemometric modelling (PCM) can take information for both ligands and targets as input to predict the affinity of their interactions, which allows generation of compounds that are promiscuous (useful for e.g., viral mutants) or selective (useful for e.g., kinase inhibitors) [34]. The Transformer can then be used to construct inverse PCM models which take the protein information as input (e.g. sequences, structures, or descriptors) to design active ligands for a given protein target without known ligands. Moreover, the Transformer can also be used for lead optimization. For instance, the input can be a “hit” already, generating “optimized” ligands, or a “lead” with side effects to produce ligands with a better ADME/tox profile.

Authors’ Contributions

XL and GJPvW conceived the study and performed the experimental work and analysis.

KY, APIJ nd HWTvV provided feedback and critical input. All authors read, commented on and approved the final manuscript.

Acknowledgements

XL thanks Chinese Scholarship Council (CSC) for funding, GJPvW thanks the Dutch Research Council and Stichting Technologie Wetenschappen (STW) for financial support (STW-Veni #14410). Thanks go to Dr. Xue Yang for verifying Table S1 and Dr. Anthe Janssen checking the convergence of t-SNE. We also acknowledge Bert Beerkens for providing the common scaffolds used to generate molecules as an example.

Competing Interests

The authors declare that they have no competing interests

References

1. P. G. Polishchuk, T. I. Madzhidov and A. Varnek, *J Comput Aided Mol Des*, 2013, **27**, 675-679.
2. P. J. Hajduk and J. Greer, *Nat Rev Drug Discov*, 2007, **6**, 211-219.
3. G. L. Card, L. Blasdel, B. P. England, C. Zhang, Y. Suzuki, S. Gillette, D. Fong, P. N. Ibrahim, D. R. Artis, G. Bollag, M. V. Milburn, S. H. Kim, J. Schlessinger and K. Y. Zhang, *Nat Biotechnol*, 2005, **23**, 201-207.
4. Y. Bian and X. S. Xie, *AAPS J*, 2018, **20**, 59.
5. J. P. Hughes, S. Rees, S. B. Kalindjian and K. L. Philpott, *Br J Pharmacol*, 2011, **162**, 1239-1249.
6. C. Sheng and W. Zhang, *Med Res Rev*, 2013, **33**, 554-598.
7. R. Santos, O. Ursu, A. Gaulton, A. P. Bento, R. S. Donadi, C. G. Bologa, A. Karlsson, B. Al-Lazikani, A. Hersey, T. I. Oprea and J. P. Overington, *Nat Rev Drug Discov*, 2017, **16**, 19-34.
8. B. B. Fredholm, *Exp Cell Res*, 2010, **316**, 1284-1288.
9. J. F. Chen, H. K. Eltzschig and B. B. Fredholm, *Nat Rev Drug Discov*, 2013, **12**, 265-286.
10. S. Moro, Z. G. Gao, K. A. Jacobson and G. Spalluto, *Med Res Rev*, 2006, **26**, 131-159.
11. W. Jespers, A. Oliveira, R. Prieto-Diaz, M. Majellaro, J. Aqvist, E. Sotelo and H. Gutierrez-de-Teran, *Molecules*, 2017, **22**.
12. X. Liu, A. P. IJzerman and G. J. P. van Westen, *Methods Mol Biol*, 2021, **2190**, 139-165.
13. Y. LeCun, Y. Bengio and G. Hinton, *Nature*, 2015, **521**, 436-444.
14. R. Gomez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernandez-Lobato, B. Sanchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, *ACS Cent Sci*, 2018, **4**, 268-276.
15. M. H. S. Segler, T. Kogej, C. Tyrchan and M. P. Waller, *ACS Cent Sci*, 2018, **4**, 120-131.
16. S.-L. Benjamin, O. Carlos, G. Gabriel L. and A.-G. Alan, *Optimizing distributions over molecular space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC)*, 2017.
17. M. Olivecrona, T. Blaschke, O. Engkvist and H. Chen, *Journal of cheminformatics*, 2017, **9**, 48.
18. T. Blaschke, J. Arus-Pous, H. Chen, C. Margreitter, C. Tyrchan, O. Engkvist, K. Papadopoulos and A. Patronov, *Journal of chemical information and modeling*, 2020, **60**, 5918-5922.
19. J. Lim, S. Y. Hwang, S. Moon, S. Kim and W. Y. Kim, *Chem Sci*, 2019, **11**, 1153-1164.
20. Y. Li, J. Hu, Y. Wang, J. Zhou, L. Zhang and Z. Liu, *Journal of chemical information and modeling*, 2020, **60**, 77-91.
21. J. Arus-Pous, A. Patronov, E. J. Bjerrum, C. Tyrchan, J. L. Reymond, H. Chen and O. Engkvist, *Journal of cheminformatics*, 2020, **12**, 38.
22. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. J. a. e.-p. Polosukhin, *Journal*, 2017, arXiv:1706.03762.
23. Y. Yang, S. Zheng, S. Su, C. Zhao, J. Xu and H. Chen, *Chem Sci*, 2020, **11**, 8312-8322.
24. X. Liu, K. Ye, H. W. T. van Vlijmen, A. P. IJzerman and G. J. P. van Westen, *Journal of cheminformatics*, 2019, **11**, 35.
25. X. Liu, K. Ye, H. W. T. van Vlijmen, M. T. M. Emmerich, I. A. P. and G. J. P. van Westen, *Journal of cheminformatics*, 2021, **13**, 85.
26. A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani and J. P. Overington, *Nucleic Acids Res*, 2012, **40**, D1100-1107.
27. J. Degen, C. Wegscheid-Gerlach, A. Zaliani and M. Rarey, *ChemMedChem*, 2008, **3**, 1503-1507.

28. PyTorch, https://pytorch.org/).
29. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. J. a. e.-p. Houlsby, *Journal*, 2020, arXiv:2010.11929.
30. G. R. Bickerton, G. V. Paolini, J. Besnard, S. Muresan and A. L. Hopkins, *Nat Chem*, 2012, **4**, 90-98.
31. Scikit-Learn: machine learning in Python, http://www.scikit-learn.org/).
32. P. Ertl and A. Schuffenhauer, *Journal of cheminformatics*, 2009, **1**, 8.
33. A. R. Solow and S. Polasky, *Environmental and Ecological Statistics*, 1994, **1**, 95-103.
34. G. J. van Westen, J. K. Wegner, P. Geluykens, L. Kwanten, I. Vereycken, A. Peeters, A. P. Ijzerman, H. W. van Vlijmen and A. Bender, *PLoS One*, 2011, **6**, e27518.

Table S1: Atoms in vocabulary for graph-based molecule generation. The column of "Symbol" is the symbol of the atom and its charge; the column of "Valence" is the value of valence of the state of each chemical element; the "Number" column stands for the index of each element in the periodic table, the last row is the unique word for each state of these elements, a combination of its valence and symbol.

Symbol	Valence	Charge	Number	Word
O	2	0	8	2O
O+	3	1	8	3O+
O-	1	-1	8	1O-
C	4	0	6	4C
C+	3	1	6	3C+
C-	3	-1	6	3C-
N	3	0	7	3N
N+	4	1	7	4N+
N-	2	-1	7	2N-
Cl	1	0	17	1Cl
S	2	0	16	2S
S	6	0	16	6S
S	4	0	16	4S
S+	3	1	16	3S+
S+	5	1	16	5S+
S-	1	-1	16	1S-
F	1	0	9	1F
I	1	0	53	1I
I	5	0	53	5I
I+	2	1	53	2I+
Br	1	0	35	1Br
P	5	0	15	5P
P	3	0	15	3P
P+	4	1	15	4P+
Se	2	0	34	2Se
Se	6	0	34	6Se
Se	4	0	34	4Se
Se+	3	1	34	3Se+
Si	4	0	14	4Si
B	3	0	5	3B
B-	4	-1	5	4B-
As	5	0	33	5As
As	3	0	33	3As
As+	4	1	33	4As+
Te	2	0	52	2Te
Te	4	0	52	4Te
Te+	3	1	52	3Te+
*	0	0	0	*

Table S2: The pseudo code for encoding the graph representation of molecules in *DrugEx v3*

```
Algorithm encoding:
    Input:
       mol: structure of the kekulized molecule
       subs: structure of the scaffolds
       vocab: vocabulary of tokens which is consisted of graph matrix
    Output:
        matrix: the n x 5 matrix to represents the molecular graph.

    # Ensure the atom of the subs are put at the start in the molecule
    mol ← RANK_ATOM_BY_SUB(mol, subs)
    sub_atoms ← GET_ATOMS (subs)
    sub_bonds ← GET_BONDS (subs)
    mol_atoms ← GET_ATOMS (mol)
    frag, grow, link ← [('GO', 0, 0, 0, 1)], [], [(0, 0, 0, 0, 0)]
    For atom in mol_atoms:
        # The bonds which connect to the atom having the index before this atom
        bonds ← GET_LEFT_BONDS (mol, atom)
        For bond in bonds:
            tk_bond ← GET_TOKEN (vocab, bond)
            other ← GET_OTHER_ATOM(mol, atom, bond)
            If IS_FIRST (bonds, bond):
                tk_atom ← GET_TOKEN (vocab, atom)
            Else:
                tk_atom ← GET_TOKEN (vocab, None)

            # The index of the scaffold in which the current atom locates
            # Its value starts from 1. If it is not in the scaffold, it will be 0
            scf ← GET_FRAG_ID (subs, atom)
            column ← (tk_atom, tk_bond, GET_INDEX (other), GET_INDEX (atom), scf)
            If other in sub_atoms and atom in sub_atoms and bond not in sub_bonds:
                Insert column to link
            Else if bond in sub_bonds:
                Insert column to frag
            Else:
                Insert column to grow
        End
    End
    Insert ('EOS', 0, 0, 0, 0) to grow
    matrix ← CONCATENATE_BY_COLUMN (frag, grow, link)
    Return matrix
```

Table S3: The pseudo code for decoding the graph representation of molecules in *DrugEx v3*

```
Algorithm decoding:
    Input:
       matrix: the n x 5 matrix to represents the molecular graph
       vocab: vocabulary of tokens which is consisted of graph matrix
    Output:
       mol: structure of the kekulized molecule
       subs: structure of the scaffolds

    mol ← new MOL ()
    subs ← new SUB ()
    For atom, bond, prev, curr, scf in matrix:
        If atom == 'EOS' or atom == 'GO':
            continue
        If atom != '*':
            a ← new Atom (GET_ATOM_SYMBOL(vocab, atom))
            SET_FORMAL_CHARGE (a, GET_CHARGE(vocab, atom))
            ADD_ATOM (mol, a)
            If scf != 0: ADD_ATOM (subs, a)
        If bond != 0:
            b ← new Bond (bond)
            ADD_BOND(mol, b)
            If frag != 0:
                ADD_BOND (subs, b)
    End

    # automatically determine the aromatic rings
    mol ← SANITIZE (mol)
    subs ← SANITIZE (subs)
    Return mol, subs
```