# Appendix A Supplementary Materials

## A.1 Notations

All notaions used in this paper are listed in Table A1.

## A.2 ScafVAE architecture

ScafVAE consists of three parts: the encoder, the decoder, and the surrogate model. Different from conventional atom-based or fragment-based methods[1, 2, 3, 4], ScafVAE offers an efficient framework for novel molecule generation based on the concept of "bond scaffold" (explained below) with a VAE-style design (Fig.1a). The encoder converts each molecule into a latent vector with an isotropic Gaussian distribution, and the decoder reconstructs the molecule from the latent vector. Instead of generating molecules with a specific fragment set, the decoder first generates a set of fragments containing only bond types (named as "bond scaffold", i.e., fragments without specifying atom types). Then, the decoder assembles these bond scaffolds and decorates their atom types to produce the valid molecules (Fig.1e). Surrogate models are employed for molecule optimization for downstream tasks, which are augmented by contrastive learning and fingerprint reconstruction. Specifically, we used the perplexity-inspired fragmentation to obtain the bond scaffolds.

### A.2.1 Perplexity-inspired fragmentation

Inspired by the widely used perplexity metric in AI language models[5], we used perplexity to break molecular bonds to obtain the fragments. In this paper, the bond perplexity is defined as the exponential of the entropy of each bond under a pre-trained masked graph model (i.e. perplexity estimator). The masked graph model is a neural network that learns to predict the actual bond type of a masked bond based on the observable atoms and bonds (Fig.1c). For a given molecular graph, we predict the perplexity of one bond at a time, meaning that we only mask a single bond while keeping the rest of the graph visible. Given a predicted distribution of a masked bond, its bond perplexity (PPL) can be written as:

$$PPL = e^{-\sum_i^{N_b} t_i log(p_i)} \tag{A1}$$

where $N_b$ is the number of bond types, $t_i$ is the true label and the $p_i$ is the predicted probability for $i^{th}$ bond type. All bonds with a perplexity greater than $c_{ppl}$ were broken ($c_{ppl}$ is a hyperparameter, and $c_{ppl} = 1.5$ in this study). To simplify the molecule reconstruction, we only break non-ring single bonds, i.e., no rings formed between fragments. Moreover, the non-ring single bond with the highest perplexity within each fragment is broken until its number of atoms is less or equal to 6. The detailed architecture and training procedure of the perplexity estimator were described in Sec.A.2.7.

### A.2.2 Encoder

Given a molecule graph, the encoder converts the molecule into a 64-D vector (Fig.1c). The all-atom molecule graph is first fed into two GNN blocks. Subsequently, the all-atom graph is converted to a coarse-grained graph by applying a local average pool according to its fragmentation (i.e., all nodes belonging to the same fragment are pooled into a single node). Then, the coarse-grained graph is fed into two RGNN blocks iteratively. After each iteration, a node is removed until no node exists. Finally, the output of RGNN blocks is used to predict the mean ($\boldsymbol{\mu}$) and variance ($\boldsymbol{\sigma}^2$) of the variational distribution over the latent space. The latent vector ($\mathbf{z}$) is sampled as $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$. In particular, node removal follows the breadth-first search (BFS) ordering, which gives a sub-optimal solution that ensures the node being removed connects to at least one of the remaining nodes, and such a strategy was wildly adopted in previous studies[1, 6, 4]. The hidden size of the node/edge feature is 128/64 for all-atom graphs and 1024/64 for coarse-grained graphs, respectively. The hidden size of the memory vector in RGNN is 2048.

### A.2.3 Decoder

The decoder reconstructs the molecule from the latent vector $\mathbf{z}$ using a network structure similar to the encoder (Fig.1e). Two RGNN blocks are first applied to iteratively generate bond scaffolds. Each iteration generates a new bond scaffold based on the existing ones. The generated bond scaffolds are then processed by an assembler, which consists of two MLPs. One of them is used to predict which two bond scaffolds can be connected (i.e., bond formation), while the other is used to predict which two atoms can form a bond between two bond scaffolds. Finally, an atom decorator, which comprises two GNN blocks, generates the atoms iteratively, with each iteration generating one atom. The hidden size of all blocks in the decoder is the same as in the encoder. The decoded molecules are post-processed by the Rectifier module in Fragmenstein[7] to fix minor invalid parts.

### A.2.4 GNN block

The GNN block used in this study is a variant of graph attention network[8], which performs massage passing with the multi-head attention mechanism. For a central node $\mathbf{x}_i$ (hidden size of $h_x$), the GNN block updates its feature with neighbor node $\mathbf{x}_j$ (all neighbor nodes are denoted as $Neighbor(i)$) and their edge $\mathbf{e}_{ij}$(hidden size of $h_e$). The GNN block can be described as follows:

**Algorithm 1:** Graph neural network (GNN) block

**Input:** Central node $\mathbf{x}_i \in \mathbb{R}^{h_x}$, neighbor node $\mathbf{x}_j \in \mathbb{R}^{h_x}, j \in Neighbor(i)$, and their edge $\mathbf{e}_{ij} \in \mathbb{R}^{h_e}$.
**Output:** Updated central node $\mathbf{x}_i^u$ and edge $\mathbf{e}_{ij}^u \in \mathbb{R}^{h_e^s}$.

**Def** GNN$(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ij}, N_h = 8)$**:**

`// Attention-based message aggregation.`

$\mathbf{q}_i^h \leftarrow \text{LinearNoBias}(\text{LayerNorm}(\mathbf{x}_i))$ $\quad \mathbf{q}_i^h \in \mathbb{R}^{h_x^a}, h \in \{1, ..., N_h\}$

$\mathbf{k}_j^h, \mathbf{v}_j^h \leftarrow \text{LinearNoBias}(\text{LayerNorm}(\mathbf{x}_j))$
$\mathbf{k}_i^h \in \mathbb{R}^{h_x^a}, \mathbf{v}_i^h \in \mathbb{R}^{h_x^a}, h \in \{1, ..., N_h\}$

$\mathbf{b}_{ij}^h \leftarrow \text{LinearNoBias}(\text{LayerNorm}(\mathbf{e}_{ij}))$ $\quad \mathbf{b}_{ij}^h \in \mathbb{R}^{h_x^a}, h \in \{1, ..., N_h\}$

$\mathbf{a}_{ij}^h \leftarrow \mathbf{q}_i^h \odot \mathbf{k}_j^h + \mathbf{b}_{ij}^h$ $\quad \mathbf{a}_{ij}^h \in \mathbb{R}^{h_x^a}$

$\hat{a}_{ij}^h \leftarrow \text{softmax}_j \frac{1}{\sqrt{h_x^a}} \mathbf{a}_{ij}^h \mathbf{1}$ $\quad \hat{a}_{ij}^h \in \mathbb{R}^1$

$\mathbf{x}_i^a \leftarrow \mathbf{x}_i + \text{Linear}(\text{concat}_h(\sum_j \hat{a}_{ij}^h \mathbf{v}_j^h)$ $\quad \mathbf{x}_i^a \in \mathbb{R}^{h_x}$

$\mathbf{e}_{ij}^a \leftarrow \mathbf{x}_i + \text{Linear}(\text{concat}_h(\mathbf{a}_{ij}^h)$ $\quad \mathbf{e}_{ij}^a \in \mathbb{R}^{h_e}$

`// Feedforward.`

$\mathbf{x}_i^u \leftarrow \mathbf{x}_i^a + \text{Linear}(\text{leaky\_relu}(\text{Linear}(\text{LayerNorm}(\mathbf{x}_i^a)))$ $\quad \mathbf{x}_i^u \in \mathbb{R}^{h_x}$

$\mathbf{e}_{ij}^u \leftarrow \mathbf{e}_{ij}^a + \text{Linear}(\text{leaky\_relu}(\text{Linear}(\text{LayerNorm}(\mathbf{e}_{ij}^a)))$ $\quad \mathbf{e}_{ij}^u \in \mathbb{R}^{h_e}$

**return** $(\mathbf{x}_i^u, \mathbf{e}_{ij}^u)$

### A.2.5 RGNN block

The RGNN block is designed to learn the encoding and decoding process of a molecular graph in a sequential manner (i.e. node removal or generation). The block consists of a GNN and a GRU module[9]. The hidden state of the GRU is first transformed into a super node using a layer norm operation and a linear layer. The super node is then added to the graph, with edges connecting it to all other nodes. The features of these new edges are also derived from the GRU's hidden state, with a layer norm operation and a linear layer. Then, the graph updates its feature by applying a GNN block. The global average pooling of the graph is then used to update the GRU's hidden state, while the GRU uses the gate mechanism to remember input information selectively.

### A.2.6 Surrogate model

The surrogate model is employed for molecule generation and optimization of downstream tasks, which tasks the latent vector as input and predicts various properties (Fig.1d). It consists of two parts: two shallow pre-trained MLPs and a task-specific ML module. Two MLPs were augmented by pre-training with contrastive learning and fingerprint reconstruction.

**Pre-trained MLP.** Two pre-trained MLPs decode the latent vector into two meaningful representations (i.e., two vectors), one for contrastive learning and another for molecular fingerprint reconstruction (Fig.1d), respectively.

**Task-specific ML module.** The ML module takes the two vectors as input and predicts specific molecular properties. In this study, we applied a range of ML algorithms, including Adaboost, SVM, KNN, RF, and MLP. All these ML modules were implemented with the Scikit-Learn library.

### A.2.7 Perplexity estimator

The perplexity estimator is a masked graph model, consisting of two GNN blocks with hidden sizes of 128 and 64 for features of nodes and edges, respectively. We trained the perplexity estimator using the ChEMBL dataset. For each training batch, we randomly masked one bond in each molecular graph. The loss was calculated as a classification task using cross-entropy loss. The model was initially trained for 1650 iterations, with a maximum number of nodes set at 64 (learning rate of $5 \times 10^{-3}$, batch size of 384). Subsequently, it underwent additional training of 490 iterations, with a maximum number of nodes increased to 128 (learning rate of $5 \times 10^{-4}$, batch size of 256).

## A.3 Supplementary Results

### A.3.1 Single-objective molecule generation

We conducted two experiments to generate single-objective molecules aimed at achieving a high QED score or a low SA score. As shown in Fig.A22a and Fig.A22b, the generated molecules outperformed those in the training set, achieving higher QED scores and lower SA scores. Notably, the model samples molecules based on predicted properties, which align with the desired actual properties observed. This suggests that sampling using predicted properties is effective.

### A.3.2 Tri-objective molecule generation

We conducted four experiments to generate tri-objective molecules aimed at achieving a stronger Vina score, a high QED score, and a low SA score, targeting EGFR (Fig.A23a-c), HER2 (Fig.A23d-f), P-gp (Fig.A23g-i) and BCRP (Fig.A23j-l). We presented the actual values of these three properties rather than the predicted properties since the actual values can be easily obtained. These results demonstrate that molecules with multiple desired properties can be sampled based on the predicted properties.

### A.3.3 Downsizing vocabulary set with bond scaffold

We evaluated the ability of bond scaffold strategy to reduce the vocabulary size of JT-VAE[1], as shown in Fig.A21. With its default fragmentation settings, JT-VAE contains 7736 fragments for the ChEMBL dataset and 780 fragments for the ZINC dataset. However, when we applied the bond scaffold strategy, the size was reduced to 2980 and 89, respectively. These indicate that the majority of the fragments share the same bond scaffold, and the bond scaffold strategy can efficiently control the vocabulary size during molecular generation.

## A.4 Supplementary Tables

**Table A1 Notations.**

| Notation | Description |
|---|---|
| $N_b$ | Number of bond types |
| $N_{aa}^e$ | Number of GNN blocks in the encoder |
| $N_{cg}^e$ | Number of RGNN blocks in the encoder |
| $N_{aa}^d$ | Number of GNN blocks in the decoder |
| $N_{cg}^d$ | Number of RGNN blocks in the decoder |
| $c_{ppl}$ | Cutoff for perplexity-inspired fragmentation |
| $N_h$ | Number of attention heads |
| $h_x$ | Hidden size of node features |
| $h_e$ | Hidden size of edge features |
| $h_x^a$ | Hidden size of node features of each attention head |
| $h_e^a$ | Hidden size of node features of each attention head |
| Linear | Linear transformation with bias |
| LinearNoBias | Linear transformation without bias |
| MLP | Multi-layer perceptron |
| LayerNorm | Layer normalization |
| $\odot$ | Hadamard product |
| $\oslash$ | Hadamard division |
| concat | Tensor concatenating |
| sigmoid | Sigmoid function |
| leaky_relu | Leaky ReLU activation function |
| softmax | Softmax function |
| $\mathbf{1}$ | Column vector of 1s |

**Table A2 Reconstruction performance of randomly sampled 10000 molecules on the ChEMBL test set.**

| Metrics | Success rate (%) |
|---|---|
| Bond scaffold reconstruction | 93.21% |
| Molecular reconstruction | 68.28% |

**Table A3 Results of GuacaMol distribution-learning benchmarks on the ChEMBL dataset.**

| Type | Model | Validity | Uniqueness | Novelty | KL divergence | Fréchet ChemNet Distance |
|---|---|---|---|---|---|---|
| SMILES | LSTM[10] | 0.959 | 1.000 | 0.912 | 0.991 | 0.913 |
| | AAE[11] | 0.822 | 1.000 | 0.998 | 0.886 | 0.529 |
| | ORGAN[12] | 0.379 | 0.841 | 0.687 | 0.267 | 0.000 |
| | VAE[13] | 0.870 | 0.999 | 0.974 | 0.982 | 0.863 |
| Graph | VGAE[14] | 0.830 | 0.944 | 1.000 | 0.554 | 0.016 |
| | Graph MCTS[15] | 1.000 | 1.000 | 0.994 | 0.522 | 0.015 |
| | VGAE-MCTS[16] | 1.000 | 1.000 | 1.000 | 0.659 | 0.009 |
| | ScafVAE | 0.987 | 1.000 | 1.000 | 0.959 | 0.338 |

*Performance data for LSTM, AAE, ORGAN, and VAE was obtained from Ref.[17]. Performance data for VGAE, Graph MCTS, VGAE-MCTS was obtained from Ref.[17].

**Table A4 Results of GuacaMol distribution-learning benchmarks on the ZINC dataset.**

| Type | Model | Validity | Uniqueness | Novelty | KL divergence | Fréchet ChemNet Distance |
|---|---|---|---|---|---|---|
| SMILES | LSTM[2] | 0.968 | 0.999 | 1.000 | - | - |
| | CVAE[13] | 0.007 | 0.675 | 1.000 | - | - |
| | GVAE[18] | 0.072 | 0.009 | 1.000 | - | - |
| Graph | JT-VAE[1] | - | 0.988 | 0.988 | 0.882 | 0.263 |
| | CGVAE[2] | 1.000 | 0.998 | 1.000 | - | - |
| | GCPN[19] | - | 0.982 | 0.982 | 0.456 | 0.003 |
| | GA[20] | - | 0.008 | 0.008 | 0.705 | 0.001 |
| | MARS[21] | - | 0.737 | 0.737 | 0.798 | 0.271 |
| | HierVAE[22] | - | 0.131 | 0.131 | 0.602 | 0.001 |
| | PS-VAE[4] | - | 0.997 | 0.997 | 0.850 | 0.318 |
| | GraphAF[23] | - | 0.288 | 0.287 | 0.508 | 0.023 |
| | GraphDF[24] | - | 0.998 | 0.998 | 0.459 | 0.001 |
| | MoFlow[25] | 1.000 | 0.999 | 1.000 | - | - |
| | GF-VAE[26] | 1.000 | 1.000 | 1.000 | - | - |
| | ScafVAE | 0.997 | 1.000 | 1.000 | 0.857 | 0.622 |

*Performance data for LSTM and CGVAE was obtained from Ref.[2]. Performance data for CVAE, GVAE and GF-VAE was obtained from Ref.[26]. Performance data for JT-VAE, GCPN, GA, MARS, HierVAE, PS-VAE, GraphAF and GraphDF was obtained from Ref.[4]. Performance data for MoFlow was obtained from Ref.[25].

**Table A5 Results of the shredding procedure for the ZINC dataset.**

| | ScafVAE | BRICS[27] |
|---|---|---|
| Cleaved compounds | 218651 (100.00%) | 216563 (99.04%) |
| Uncleaved compounds | 0 (0.00%) | 2088 (0.96%) |
| Unique fragments | 95688 (100.00%) | 34063 (100.00%) |
| 1-connection fragments | 16494 (17.24%) | 19329 (56.74%) |
| 2-connection fragments | 39783 (41.58%) | 10436 (30.64%) |
| 3-connection fragments | 30272 (31.64%) | 1988 (5.84%) |
| 4-connection fragments | 7642 (7.99%) | 213 (0.63%) |
| 5-connection fragments | 1242 (1.30%) | 8 (0.02%) |
| 6-connection fragments | 236 (0.25%) | 1 (< 0.01%) |
| 7-connection fragments | 11 (0.01%) | - |
| 8-connection fragments | 6 (0.01%) | - |
| 9-connection fragments | 1 (< 0.01%) | - |
| 10-connection fragments | 1 (< 0.01%) | - |
| Unique fragments (w/o attachment points) | 6010 | 23116 |

**Table A6 ADMET dataset sizes.**

| Dataset | Size |
|---|---|
| HIA_Hou | 404 |
| Caco2_Wang | 625 |
| Lipophilicity_AstraZeneca | 3340 |
| Solubility_AqSolDB | 7472 |
| HydrationFreeEnergy_FreeSolv | 490 |
| BBB_Martins | 1430 |
| PPBR_AZ | 1260 |
| VDss_Lombardo | 738 |
| Half_Life_Obach | 425 |
| Clearance_Hepatocyte_AZ | 918 |
| CYP2C19_Veith | 10033 |
| CYP2D6_Veith | 10390 |
| CYP3A4_Veith | 9750 |
| CYP1A2_Veith | 9947 |
| CYP2C9_Veith | 9952 |
| hERG | 490 |
| AMES | 4418 |
| DILI | 343 |
| Skin reaction | 332 |
| LD50_Zhu | 6163 |

**Table A7 Protein-ligand binding dataset sizes.**

| Protein target | Number of binders |
|---|---|
| EGFR | 24403 |
| HER2 | 6150 |
| P-gp | 2417 |
| BCRP | 1602 |
| PARP | 1551 |
| PI3K | 12024 |
| HDAC | 3357 |
| BRD4 | 10842 |

**Table A8 Hyperparameter search space of task-specific ML module considered for ADMET tasks.**

| ML algorithm | Task type | Hyperparameter | Search space |
|---|---|---|---|
| Adaboost | Classification | n_estimators | {50, 100, 200} |
| | | learning_rate | {0.01, 0.1, 1} |
| | | algorithm | {SAMME, SAMME.R} |
| | Regression | n_estimators | {50, 100, 200} |
| | | learning_rate | {0.01, 0.1, 1} |
| | | loss | {linear, square} |
| SVM | Classification | penalty | {l1, l2} |
| | | loss | {squared_hinge, hinge} |
| | | C | {0.1, 1, 10, 100} |
| | | max_iter | {1000, 10000} |
| | Regression | epsilon | {0, 0.5, 1, 5, 10} |
| | | loss | {epsilon_insensitive, squared_epsilon_insensitive} |
| | | C | {0.1, 1, 10, 100} |
| | | max_iter | {1000, 10000} |
| KNN | Classification | n_neighbors | {1, 5, 9} |
| | | weights | {uniform, distance} |
| | | leaf_size | {10, 30, 50} |
| | Regression | n_neighbors | {1, 5, 9} |
| | | weights | {uniform, distance} |
| | | eaf_size | {10, 30, 50} |
| MLP | Classification | hidden_layer_sizes | {(100, 100), (300, 300)} |
| | | activation | {tanh, relu} |
| | | solver | {adam} |
| | | max_iter | {100, 300} |
| | Regression | hidden_layer_sizes | {(100, 100), (300, 300)} |
| | | activation | {tanh, relu} |
| | | solver | {adam} |
| | | max_iter | {100, 300} |
| RF | Classification | n_estimators | {50, 100, 200} |
| | | max_depth | {5, 40} |
| | | max_features | {auto, sqrt} |
| | | criterion | {gini, entropy} |
| | Regression | n_estimators | {50, 100, 200} |
| | | max_depth | {5, 40} |
| | | max_features | {auto, sqrt} |
| | | criterion | {squared_error, friedman_mse, poisson, absolute_error} |

**Table A9 Weights used for selecting molecules from the Pareto front of the generated multi-objective compounds with the pseudo-weight vector approach.**

| Objective | Weight |
|---|---|
| HDAC docking score | 0.05 |
| BRD4 docking score | 0.05 |
| QED score | 0.08 |
| SA score | 0.05 |
| hERG inhibition | 0.07 |
| CYP2C19 inhibition | 0.07 |
| CYP2D6 inhibition | 0.07 |
| CYP3A4 inhibition | 0.07 |
| CYP1A2 inhibition | 0.07 |
| CYP2C9 inhibition | 0.07 |
| AMES mutagenicity | 0.07 |
| Drug-induced liver injury | 0.07 |
| Skin reaction | 0.07 |
| LD50 | 0.07 |
| solubility | 0.07 |

**Table A10 Time cost of ScafVAE.**

| ScafVAE | Time cost (training) | Time cost (inference) |
|---|---|---|
| $\text{Encoder}^{\text{ChEMBL}}$ + $\text{Decoder}^{\text{ChEMBL}}$ | ≈ 3d23h | $1.58 \times 10^{-1}$s |
| $\text{Decoder}^{\text{ChEMBL}}$ | - | $4.18 \times 10^{-2}$s |
| $\text{Encoder}^{\text{ZINC}}$ + $\text{Decoder}^{\text{ZINC}}$ | ≈ 2d14h | $1.32 \times 10^{-1}$s |
| $\text{Decoder}^{\text{ZINC}}$ | - | $3.48 \times 10^{-2}$s |
| Surrogate model (Adaboost) | 32.73min | $3.18 \times 10^{-2}$s |
| Surrogate model (SVM) | 11.34min | $9.17 \times 10^{-5}$s |
| Surrogate model (KNN) | 0.32min | $2.89 \times 10^{-3}$s |
| Surrogate model (RF) | 12.62min | $2.53 \times 10^{-4}$s |
| Surrogate model (MLP) | 6.69min | $9.71 \times 10^{-3}$s |

*The labels $^{\text{ChEMBL}}$ and $^{\text{ZINC}}$ indicate that the model was trained on the corresponding dataset.
*The model for the ChEMBL dataset was trained with 8 NVIDIA RTX A40 GPU devices (memory usage ≈ 40GB per GPU), and the model for the ZINC dataset was trained with 4 NVIDIA RTX A4000 GPU devices (memory usage ≈ 13GB per GPU).
*The inference time for the encoder and decoder was tested on an NVIDIA RTX A6000 GPU device with 50000 randomly selected molecules with a batch size of 1. The average time cost was then reported.
*The training time for surrogate models was evaluated using 16 Intel(R) Xeon(R) Silver 4210R CPU cores on the CYP2D6_Veith task, which has the largest number of molecules ($N = 10390$) among all the ADMET tasks. Five conventional ML algorithms were tested with the latent space trained with the ChEMBL dataset. The inference time cost was reported as the average time taken for a batch size of 1 on a single CPU core.

## A.5 Supplementary Figures

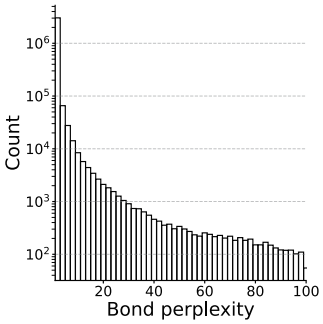


**Fig. A1 Distribution of bond perplexity in the training set.**

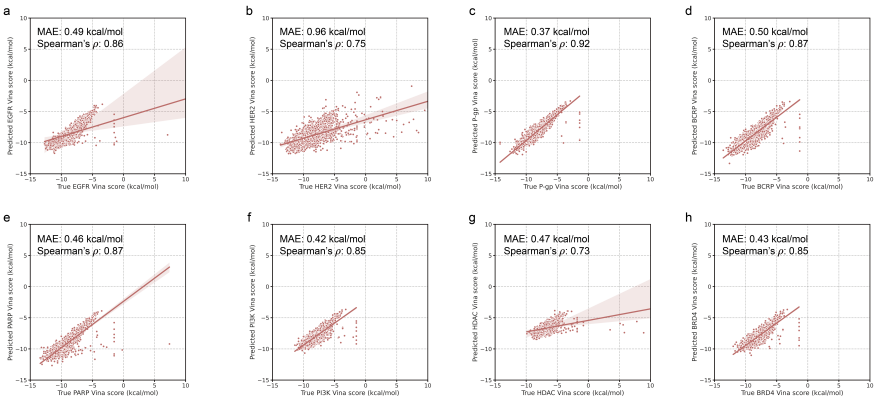


**Fig. A2 Performance of ScafVAE in predicting docking score of eight target proteins on the test set.** MAE and Spearman's $\rho$ values were shown for each protein.

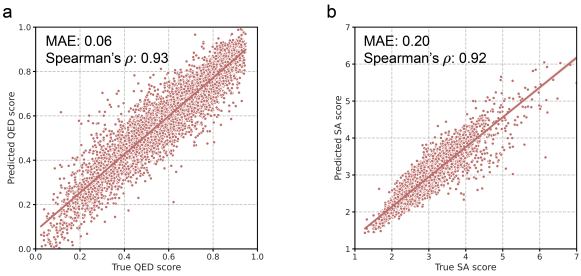


**Fig. A4 Performance of ScafVAE in predicting QED and SA score on the test set.** MAE and Spearman's $\rho$ values of each protein were shown.

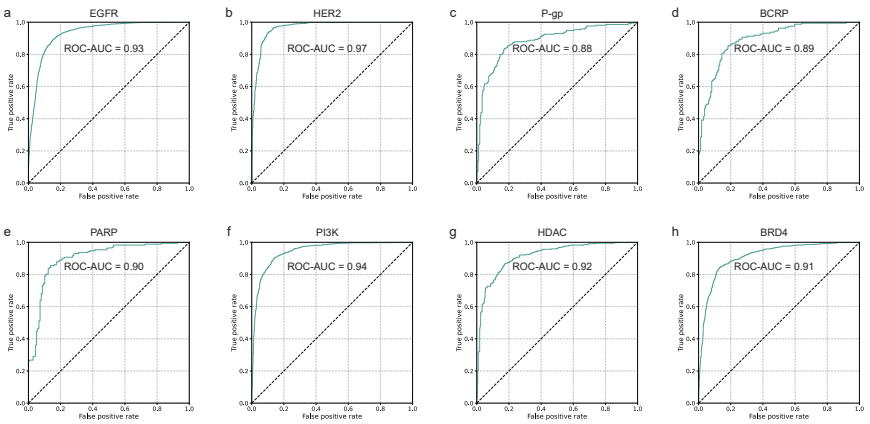


**Fig. A3 Performance of ScafVAE in classifying inhibitors of eight target proteins on the test set.** ROC-AUC values were shown for each protein.

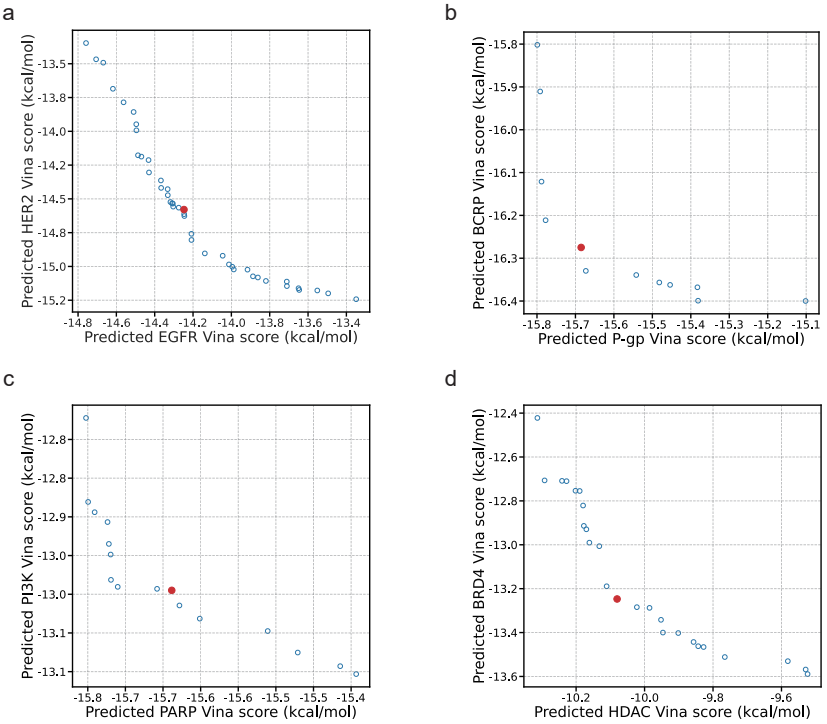


**Fig. A5 Visualization of representative Pareto front of docking score-based generation for the four resistance mechanisms.** The selected molecule from the Pareto front using the pseudo-weight vector approach was highlighted in red.

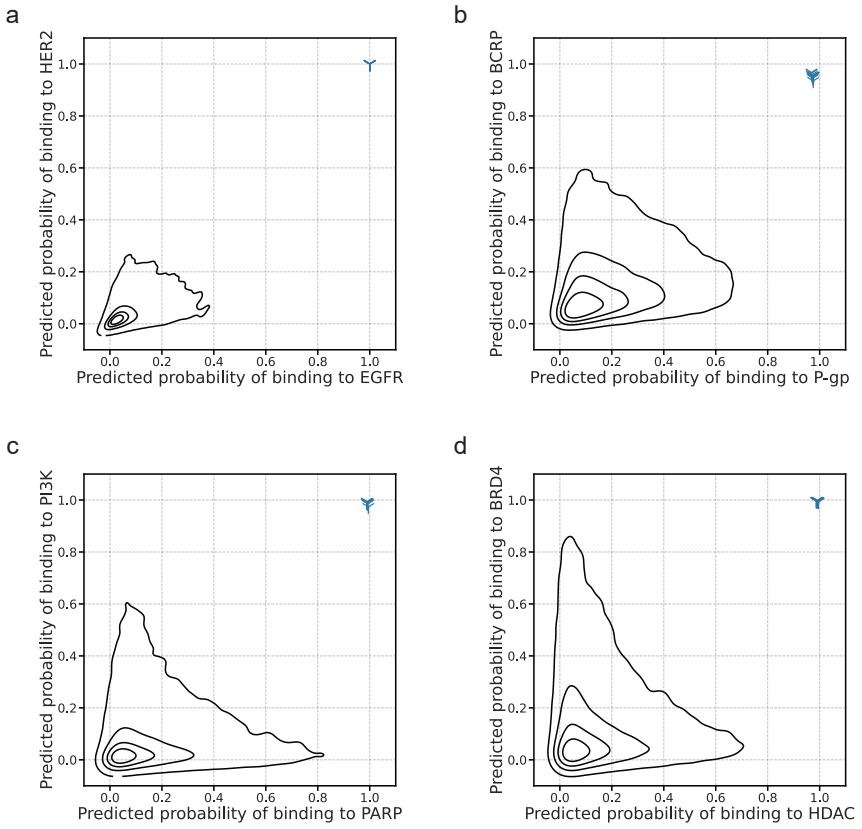


**Fig. A6 Visualization of predicted probability of binding to targets of ten generated molecules in binding probability-based generation for four resistance mechanisms.** The predicted probability of generated molecules was shown in blue, and the distribution of those in the training set was shown in black.

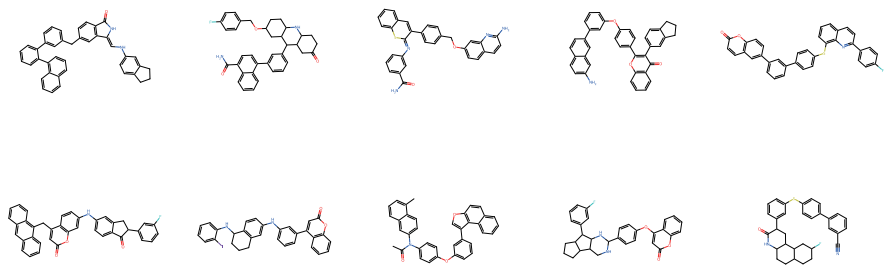

**Fig. A7 Molecules generated against EGFR and HER2 with docking score-based generation.**

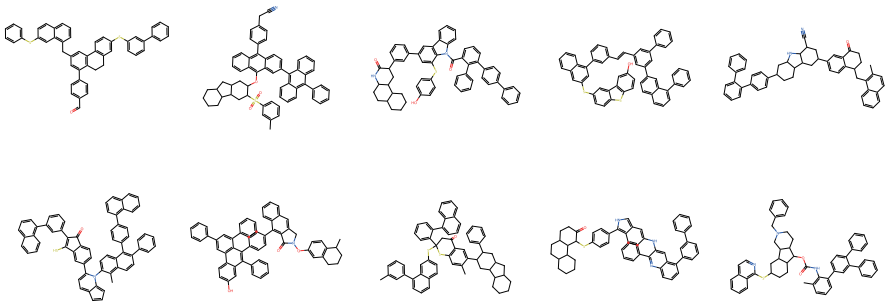

**Fig. A8 Molecules generated against P-gp and BCRP with docking score-based generation.**

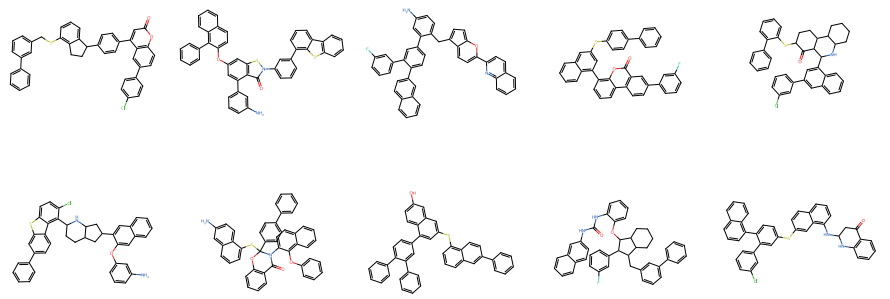

**Fig. A9 Molecule generated against PARP1 and PI3K with docking score-based generation.**

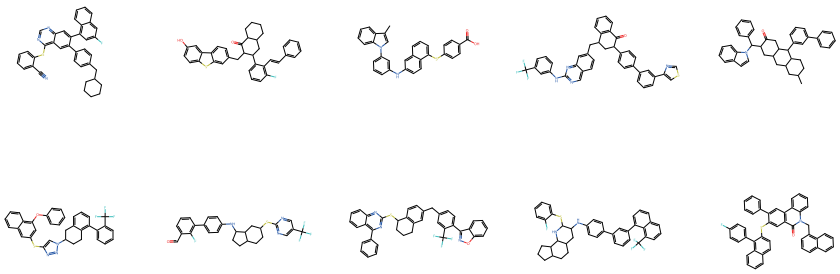

**Fig. A10 Molecules generated against HDAC and BRD4 with docking score-based generation.**

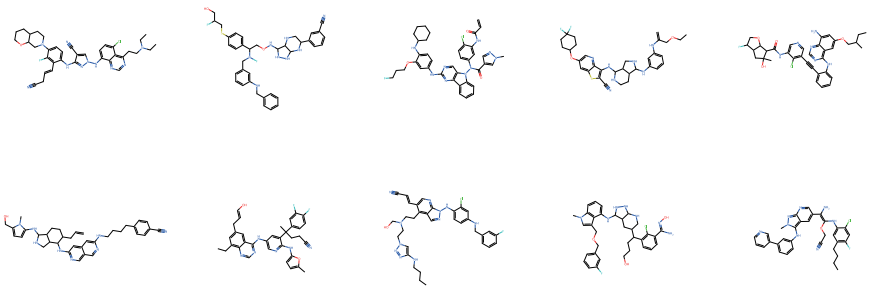

**Fig. A11 Molecules generated against EGFR and HER2 with binding probability-based generation.**

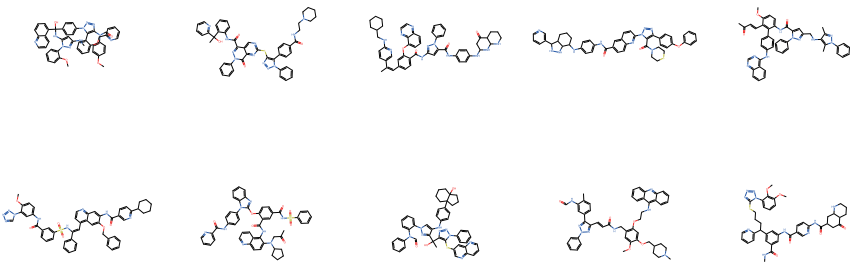

**Fig. A12 Molecules generated against P-gp and BCRP with binding probability-based generation.**

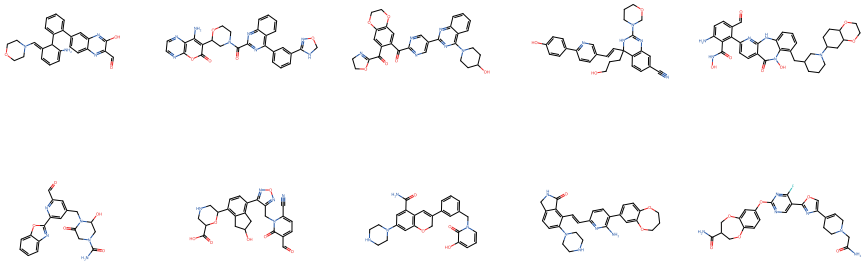

**Fig. A13 Molecule generated against PARP1 and PI3K with binding probability-based generation.**

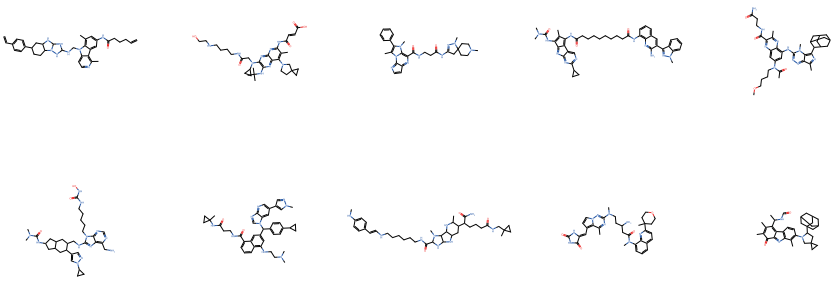

**Fig. A14 Molecules generated against HDAC and BRD4 with binding probability-based generation.**

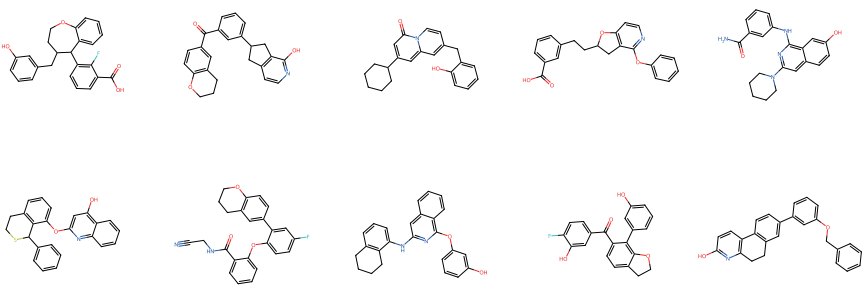

**Fig. A15 Molecules generated against EGFR and HER2 with docking score-based generation, with additional optimization of the QED score.**

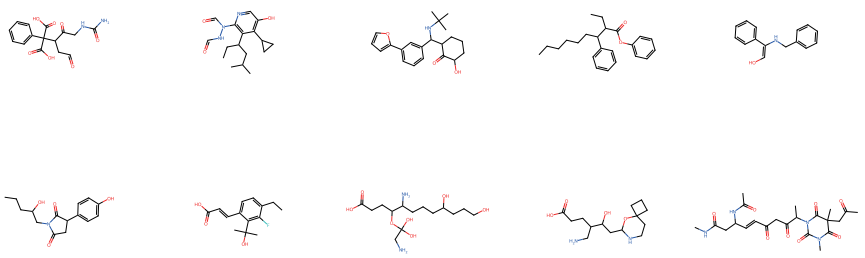

**Fig. A16 Molecules generated against HDAC and BRD4 with docking score-based generation, with additional optimization of the QED score, hERG inhibition, CYP2C19 inhibition, CYP2D6 inhibition, CYP3A4 inhibition, CYP1A2 inhibition, CYP2C9 inhibition, AMES mutagenicity, drug-induced liver injury, skin reaction, LD50,and solubility.**

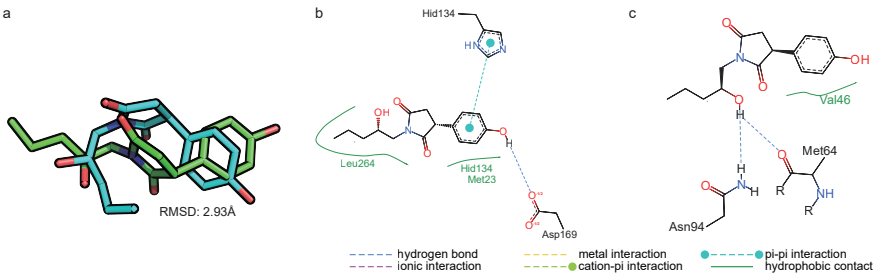


**Fig. A17 Visualization of MD simulation frame shots of compound 3 with HDAC and BRD4.** (**a**) Frame shots of compound 3 in MD simulation with HDAC (green, frame shot at 8.28ns) and BRD4 (cyan, frame shot at 3.46ns). The oxygen and nitrogen atoms were colored red and blue, respectively. (**b-c**) Visualization of protein-ligand interaction in frame shots of compound 3 with HDAC(**b**) and BRD4(**c**), respectively.

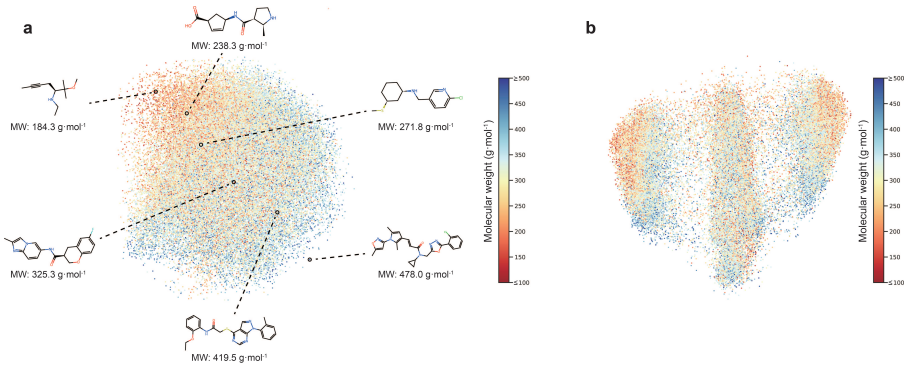


**Fig. A18 Visualization of latent space learned by ScafVAE (a) and JT-VAE (b) via t-SNE on the ZINC dataset, respectively.**

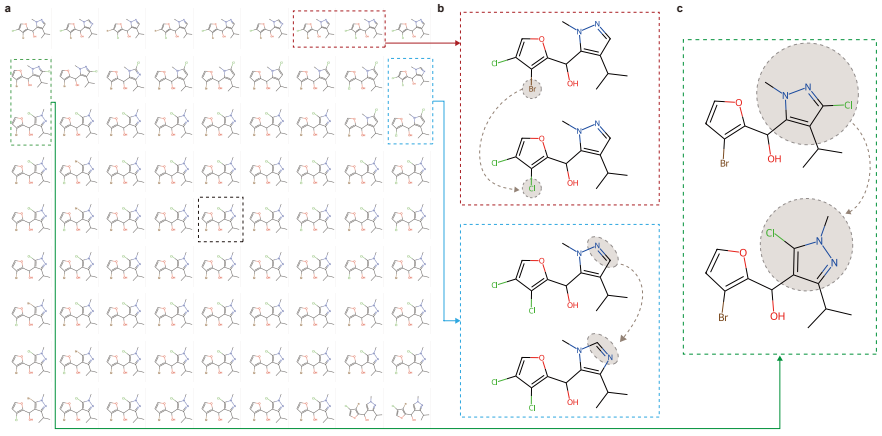


**Fig. A19 Visualization of the local neighborhood of a molecule in the center.** Molecules highlighted in the dashed box share the same (**b**) or different (**c**) bond scaffolds.

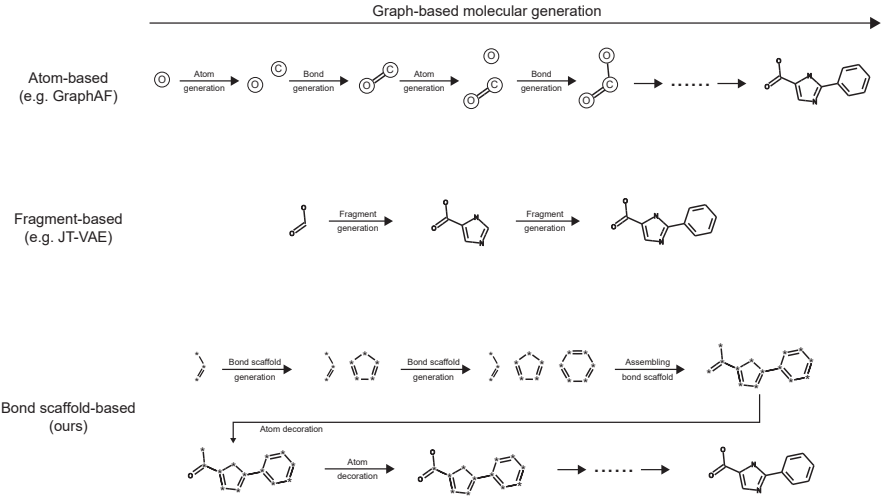


**Fig. A20 Illustration of bond scaffold generation process compared with representative atom and fragment-based methods.**

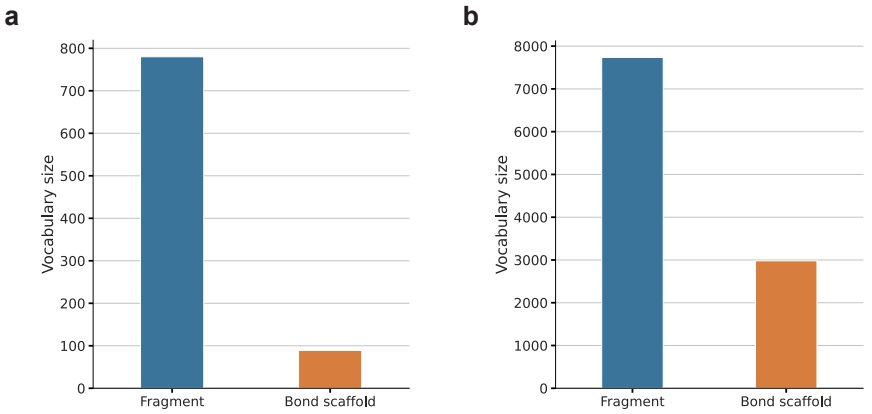


**Fig. A21 Comparison of vocabulary size using JT-VAE with and without the bond scaffold on the ZINC (a) and ChEMBL (b) dataset, respectively.**

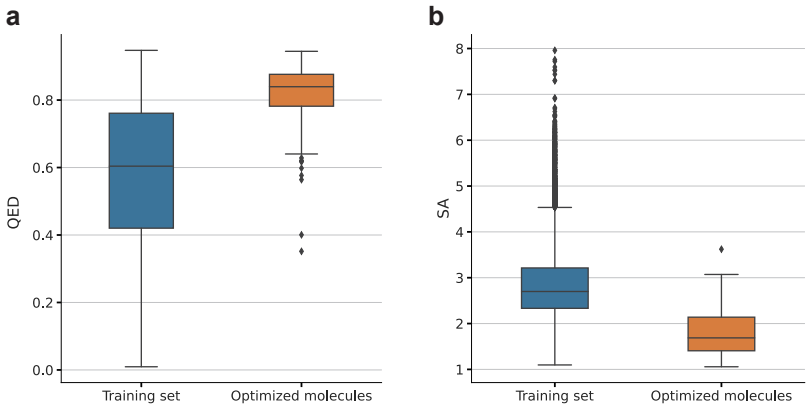


**Fig. A22 Actual properties between the training set and generated single-objective molecules for QED (a) and SA (b) score.**

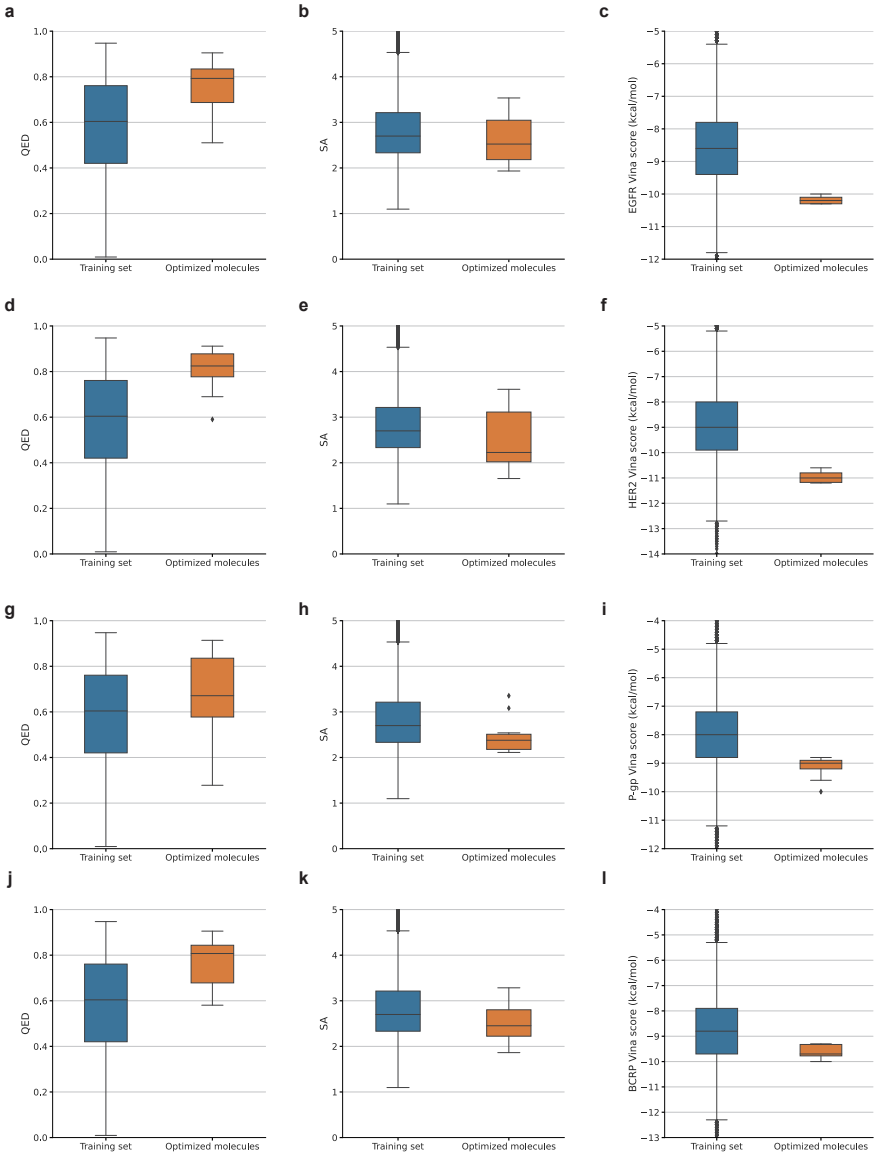


**Fig. A23 Comparison of actual properties between the training set and generated tri-objective (QED, SA, and Vina score) molecules targeting EGFR (a-c), HER2 (d-f), P-gp (d-f) and BCRP (d-f).**

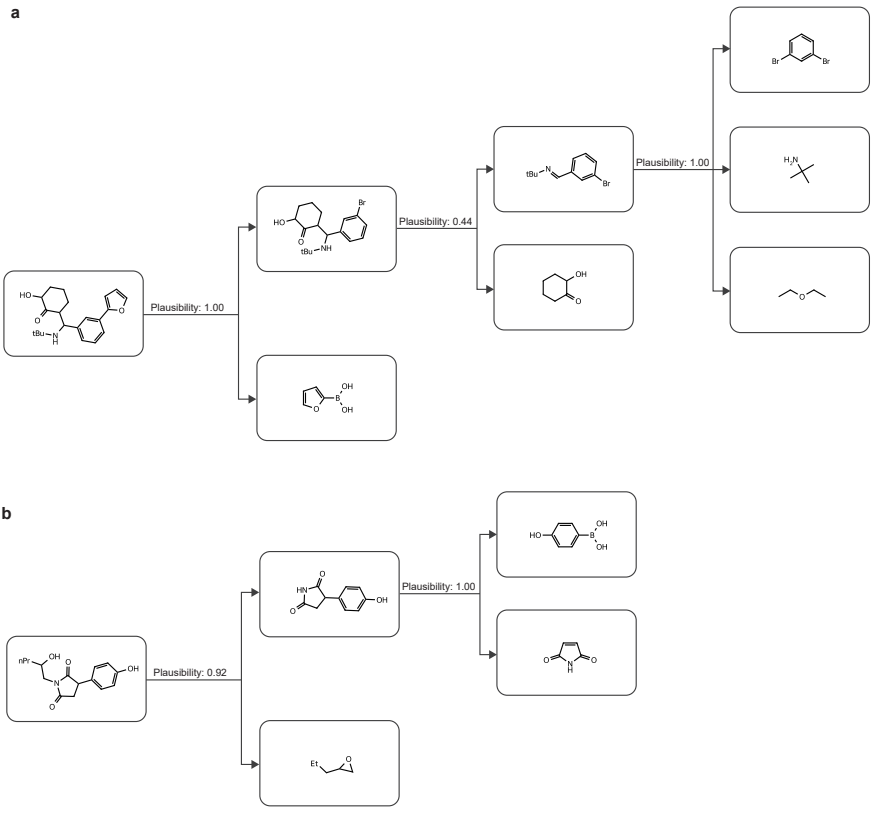


**Fig. A24 Multistep retrosynthesis planning for compound 3 (a) and 4 (b), respectively.**

# References


1. W. Jin, R. Barzilay and T. Jaakkola, 2018.
2. Q. Liu, M. Allamanis, M. Brockschmidt and A. Gaunt, Constrained graph variational autoencoders for molecule design, *Advances in neural information processing systems*, 2018, **31**.
3. P. Polishchuk, CReM: chemically reasonable mutations framework for structure generation, *J. Cheminf.*, 2020, **12**, 28.
4. X. Kong, W. Huang, Z. Tan and Y. Liu, Molecule generation by principal subgraph mining and assembling, *Advances in Neural Information Processing Systems*, 2022, **35**, 2550-2563.
5. S. F. Chen, D. Beeferman and R. Rosenfeld, Evaluation metrics for language models, 1998.
6. M. Podda, D. Bacciu and A. Micheli, 2020.
7. M. P. Ferla, R. Sánchez-García, R. E. Skyner, S. Gahbauer, J. C. Taylor, F. von Delft, B. D. Marsden and C. M. Deane, Fragmenstein: predicting protein–ligand structures of compounds derived from known crystallographic fragment hits using a strict conserved-binding–based methodology, *J. Cheminf.*, 2025, **17**, 4.
8. P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, Graph attention networks, *stat*, 2017, **1050**, 10-48550.
9. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, *arXiv preprint arXiv:1406.1078*, 2014.
10. D. Polykovskiy, A. Zhebrak, D. Vetrov, Y. Ivanenkov, V. Aladinskiy, P. Mamoshina, M. Bozdaganyan, A. Aliper, A. Zhavoronkov and A. Kadurin, Entangled conditional adversarial autoencoder for de novo drug discovery, *Mol. Pharmaceutics*, 2018, **15**, 4398-4405.
11. A. Graves and A. Graves, Long short-term memory, *Supervised sequence labelling with recurrent neural networks*, 2012, 37-45.
12. G. L. Guimaraes, B. Sanchez-Lengeling, C. Outeiral, P. L. C. Farias and A. Aspuru-Guzik, Objective-reinforced generative adversarial networks (organ) for sequence generation models, *arXiv preprint arXiv:1705.10843*, 2017.
13. R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, Automatic chemical design using a data-driven continuous representation of molecules, *ACS Cent. Sci.*, 2018, **4**, 268-276.
14. Y. Kwon, J. Yoo, Y. Choi, W. Son, D. Lee and S. Kang, Efficient learning of non-autoregressive graph variational autoencoders for molecular graph

generation. J Cheminform 11 (1): 70.*Journal*, 2019.

15. J. H. Jensen, A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space, *Chem. Sci.*, 2019, **10**, 3567-3572.
16. H. Iwata, T. Nakai, T. Koyama, S. Matsumoto, R. Kojima and Y. Okuno, VGAE-MCTS: a new molecular generative model combining the variational graph auto-encoder and monte carlo tree search, *J. Chem. Inf. Model.*, 2023, **63**, 7392-7400.
17. N. Brown, M. Fiscato, M. H. Segler and A. C. Vaucher, GuacaMol: benchmarking models for de novo molecular design, *J. Chem. Inf. Model.*, 2019, **59**, 1096-1108.
18. M. J. Kusner, B. Paige and J. M. Hernández-Lobato, 2017.
19. J. You, B. Liu, Z. Ying, V. Pande and J. Leskovec, Graph convolutional policy network for goal-directed molecular graph generation, *Advances in neural information processing systems*, 2018, **31**.
20. A. Nigam, P. Friederich, M. Krenn and A. Aspuru-Guzik, Augmenting genetic algorithms with deep neural networks for exploring the chemical space, *arXiv preprint arXiv:1909.11655*, 2019.
21. Y. Xie, C. Shi, H. Zhou, Y. Yang, W. Zhang, Y. Yu and L. Li, Mars: Markov molecular sampling for multi-objective drug discovery, *arXiv preprint arXiv:2103.10432*, 2021.
22. W. Jin, R. Barzilay and T. Jaakkola, 2020.
23. C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang and J. Tang, Graphaf: a flow-based autoregressive model for molecular graph generation, *arXiv preprint arXiv:2001.09382*, 2020.
24. Y. Luo, K. Yan and S. Ji, 2021.
25. C. Zang and F. Wang, 2020.
26. C. Ma and X. Zhang, 2021.
27. J. Degen, C. Wegscheid-Gerlach, A. Zaliani and M. Rarey, On the art of compiling and using'drug-like'chemical fragment spaces, *ChemMedChem*, 2008, **3**, 1503.