GexMolGen: Cross-modal Generation of Hit-like Molecules via Large Language Model Encoding of Gene Expression Signatures

Jiabei Cheng[1], Xiaoyong Pan[1], Kaiyuan Yang[1], Shenghao Cao[1], Bin Liu[1], Qingran Yan[2*], Ye Yuan[1*]

[1*]Department of Automation, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China.

[2*]Department of Rheumatology, Ren Ji hospital, Shanghai Jiaotong University School of Medicine, Shanghai, 200127, China.

*Corresponding author(s). E-mail(s): yanqingran@renji.com; yuanye_auto@sjtu.edu.cn;

Abstract

The design of custom drugs with specific biological activity is a extremely difficult task, but it holds the potential to generate molecules without the discovery of target genes, beyond the modern paradigm of drug discovery. Traditional search methods rely heavily on existing recorded perturbation experiments, which is costly and lacks generalization. To overcome this limitation, we propose GexMolGen (Gene Expression-based Molecule Generator), a novel model that generates hit-like molecules using gene expression signatures derived from both the initial and desired states of gene expression. Our approach follows a "first-align-then-generate" strategy where we align the gene expression signatures and molecules within a mapping space, enabling a smooth transition from the former to the latter. The transformed molecular embeddings are then decoded to molecule graphs. In this framework, we employ an advanced single-cell large language model to allow flexibility in genetic modal input. We also pre-train a scaffold-based molecule model to improve efficiency, by guaranteeing all generated molecules are 100% valid. Empirical studies demonstrate that our model outperforms traditional search methods and offers unique advantages over existing deep learning methods. Overall, our model attempts to explore the chemical and biological correlations in order to facilitate precision medicine. The usage code for GexMolGen have been released on https://github.com/Bunnybeibei/GexMolGen.

1 Introduction

Designing new molecules based on genetic information is a challenging but promising task [1]. Traditional search methods like CMAP [2] rely on comparing observed gene expression changes, referred to as gene expression signatures, with those recorded in molecular perturbation experiments. This leads to a highly restricted search space that cannot accommodate the vast possibilities within the molecular space.

There are several emerging in silico methods attempting to overcome this shortcoming. Some models, like DLEPS [3], use an indirect approach by first predicting gene expression signatures as the effects of drug candidates. Then these signatures are utilized to search for datasets in order to screen potential molecules. Other methods, like WGAN [4] and Gex2SGen [5], directly input gene expression signatures and generate candidates. WGAN uses generative adversarial networks (GAN), while Gex2SGen uses variational autoencoders (VAE). However, these models still have limitations in generalization. For example, their users have to input gene expression signatures that match the format used during the training phase, otherwise, re-training is required. Additionally, they mostly use simplified molecular-input line-entry system (SMILES) to represent molecules, which lacks structural information, leads to confusion in multiple representations, and is sensitive to the different choice of SMILES [6]. Furthermore, the mechanism from the genetic modal to the molecular modal is hard to explore [7], hindering further analysis. Therefore, it is necessary to explore new form of modal encoders and modal interaction ways.

Considering the remarkable success of foundation models, which have revolutionized various fields including transcriptomics, single-cell large language models hold great potential in this specific field, for example, serving as genetic encoders. These models are pre-trained on large, unlabeled single-cell datasets and then fine-tuned for specific tasks [8]. High-quality models such as scBERT [9], Geneformer [10], TOSICA [11], scGPT [12], scFoundation [13], SCimilarity [14], and GeneCompass [15] are capable of receiving the flexible input form of gene expression signatures due to their powerful generalization abilities.

For molecular encoders, there are various graph-based approaches. For example, molecular autoencoders represent a molecule as a graph, where vertices correspond to atoms and edges represent chemical bonds [16, 17]. These methods intuitively capture the shape of molecules but often lack efficiency due to low validity in generations. Another line of research involves flow-based methods like MolFlow [18], which achieve a perfect reconstruction rate but can be very computationally expensive for long chains. Scaffold-based approaches, such as hierVAE [19], utilize scaffolds as building blocks instead of individual atoms and edges, ensuring both high validity and modest computational cost.

Regarding modal interaction, the "First align then generate" strategy, inspired by models like DALL.E [20, 21], has gained great interest in biology. Recent work, including text-molecule matching [22], text-protein sequence generation [23], and PLIP [24], has shown impressive results.

Based on these ideas, we propose GexMolGen, a model that utilizes gene expression signatures to guide the generation of hit-like molecules. Specifically, we leverage an advanced single-cell foundation model scGPT to support flexibility in genetic inputs.

We pre-train hierVAE to encode molecules in a scaffold-based manner. We then align genes and molecules in a mapping space and facilitate transitions in genetic embeddings to molecular embeddings, finally generating molecule candidates. GexMolGen also enables clear measurement of modal similarity in the alignment space for further mechanism exploration. Our model demonstrates competitive or even superior performance compared to traditional search methods. It offers flexible genetic inputs and 100% validity compared to other deep learning models. Overall, we hope that our model to be a practical tool in emerging precision medicine.

2 Results

2.1 Overview of GexMolGen

GexMolGen is a general framework capable of generating hit-like molecules from two specific gene expression inputs: the initial states and the desired or post-perturbed states. The underlying principle is that molecular structure is correlated with its resulting differential gene expression, that is, gene expression signatures. GexMolGen adopts a strategy of "first align, then generate" for cross-modal generation. As shown in Fig.1A, the pipeline generally consists of 4 steps.

The 1st step involves the encoding of each modality. For the genetic encoder (visualized in Fig.1B), we apply an accessible single-cell foundation model, scGPT [12], to encode pre and post-perturbation gene expressions. We then calclulate the difference within the gene embedding space to represent expression signatures. For the molecular encoder (illustrated in Fig.1C), we pre-train a scaffold-based molecular generator [19] on the ChEMBL [25] dataset and then use its encoder to produce molecular embeddings, thereby capturing structural information more efficiently than SMILES-based encoders [3, 4].

The 2nd step, as presented in Fig.1A, is to fine-tune each encoder through aligning. As hard negative examples are crucial to final alignment effects, we employ an auxiliary-guidance approach to contrastive learning with iteratively refined granularity.

The alignment operation projects gene expression signatures and molecules into a unified space, facilitating the transition from genetic embeddings to molecular ones. The 3rd step, as depicted in Fig.1D, is implemented by a module entitled "Facilitator", which is trained using the teacher-forcing technique and subsequently applied in an auto-regressive manner. Molecular embeddings are discretized through PCA and binning. Beam search is adopted in this module to optimize the diversity of the generation.

Finally, tokens are reordered and made continuous through retrieving and inverse PCA to molecular embeddings. They are then fed into the molecular decoder (as shown in Fig.1E) to generate molecular graphs. During the training phase, this module need to compute the reconstruction loss to form a closed training loop within the entire framework.

The necessity for the molecular encoder and alignment phase is limited to the training phase only. For users, the process is simplified - all they need to do is input the initial and the desired gene expressions, and specify the quantity of molecular

candidates they want. To the best of our knowledge, GexMolGen is the first work that applies large language models to cross-modally generate molecules.

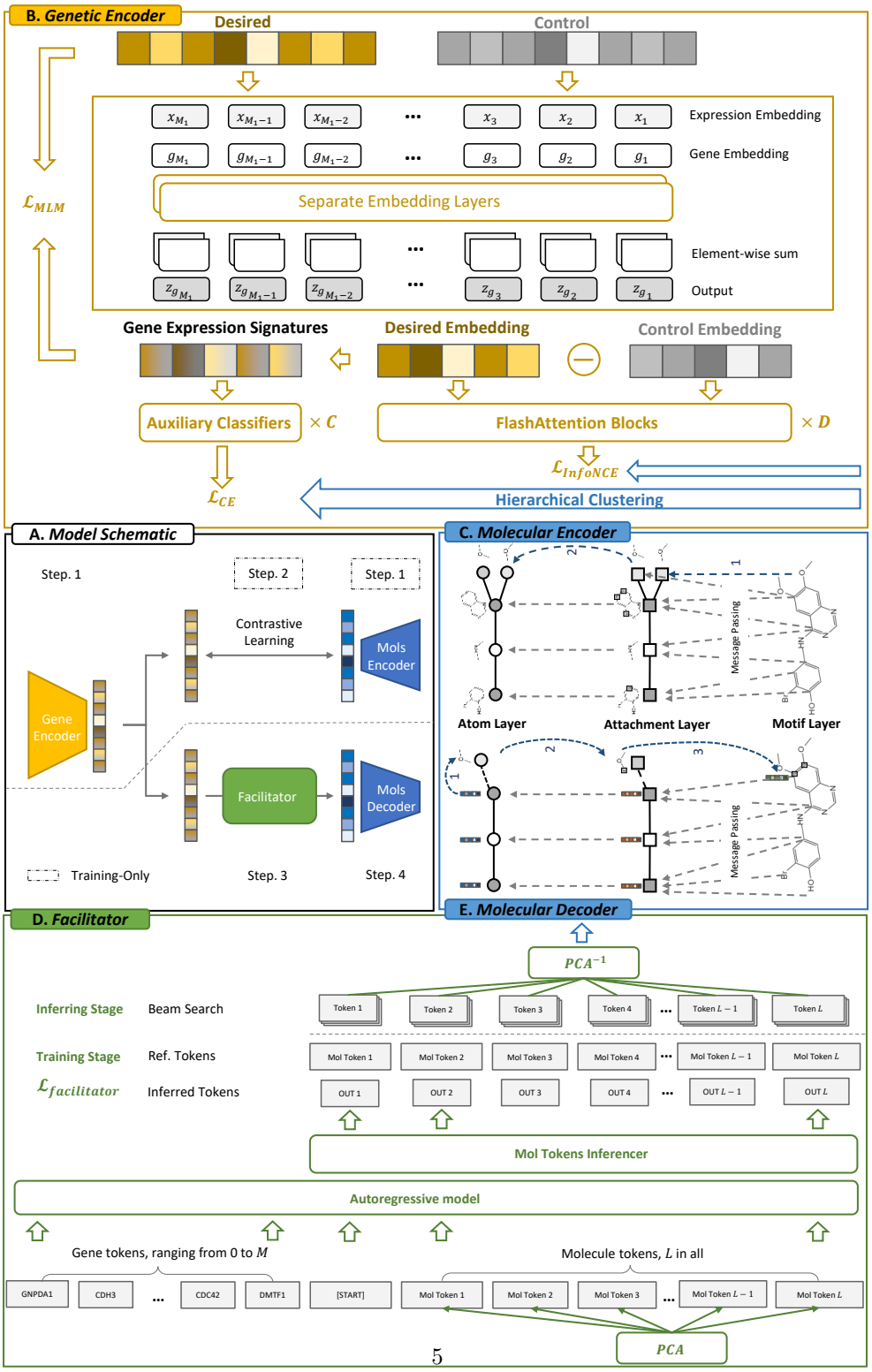


Fig. 1 The workflow of GexMolGen employs a "first-align-then-generate" strategy (Section A), which is generally divided into 4 steps: Step 1. Encoding of gene (Section B) and small molecule (Section C) data, respectively. Step 2. Aligning genetic modalities with molecular modalities (Section A). Step 3. Transformation from genetic modality to molecular modality (Section D). Step 4. Decoding to yield molecule candidates (Section E).

2.2 Generating Hit-like Molecules from Compound-Induced Gene Expression Signatures

In this section, we assess the performance of our model in generating hit-like molecules using compound-induced gene expression signatures. We input a set of N_3 hold-out samples into the model and compare the molecules generated with the actual ones in hold-out samples. In Fig.2, we illustrate this comparison (for more details, please refer to Table A.2). The results indicate that the generated molecules can successfully reproduce some complex scaffold structures appeared in the reference molecules, such as long chains (see Fig.2(4)(10)), cyclic structures (see Fig.2(1)), and cross-like structures (see Fig.2(7)).

We compare our model with two baseline methods: searching the most similar molecule in the training set according to the cosine similarity and Euclidean distance of the gene expression signatures [4]. To demonstrate the guided influence of genetic information on molecular generation, we also include a random model that generates directly from samples in the molecular latent space. For the global similarity metric, we adopt Tanimoto similarity based on Morgan fingerprints (radius=3, 1024 bits). For the local structural metric, we employ Tanimoto similarity based on MACCS keys and Fraggle similarity.

Fig.3 shows the distribution of these metrics for different methods. Detailed data can be found in Table A.2, A.3, A.4. The results obtained from our model demonstrate a significant improvement over the other three methods (p-value <0.001, according to a one-sided Mann-Whitney U-test). Specifically, our model achieves a mean Fraggle score of 0.57, a mean Morgan-based Tanimoto score of 0.30, and a mean MACCS-based Tanimoto score of 0.70. These results indicate that the molecules generated by our model exhibit a high resemblance to the reference molecules in both fingerprints and substructures. Interestingly, in Fig.3(b), our model shows a pattern that is similar to that of the cosine similarity search method. This could be explained by their common use of cosine distance as the measure of similarity between different modalities. Moreover, across all three metrics, the Euclidean distance search method consistently underperforms compared to the cosine similarity method. This discrepancy can be attributed to the curse of dimensionality, where the Euclidean distance becomes less sensitive to differences in gene expression signatures as the dimensionality increases. We also notice that the randomly generated model has the poorest performance overall. Although it occasionally produces molecules with high similarity scores (e.g. 1.19% with a similarity >0.29 in Fig.3(b)), it also generates a larger proportion of molecules that are irrelevant (e.g. 26.27% with a similarity <0.08 in Fig.3(b)). This highlights the significance of guidance from gene expression signatures in generating desirable molecular properties.

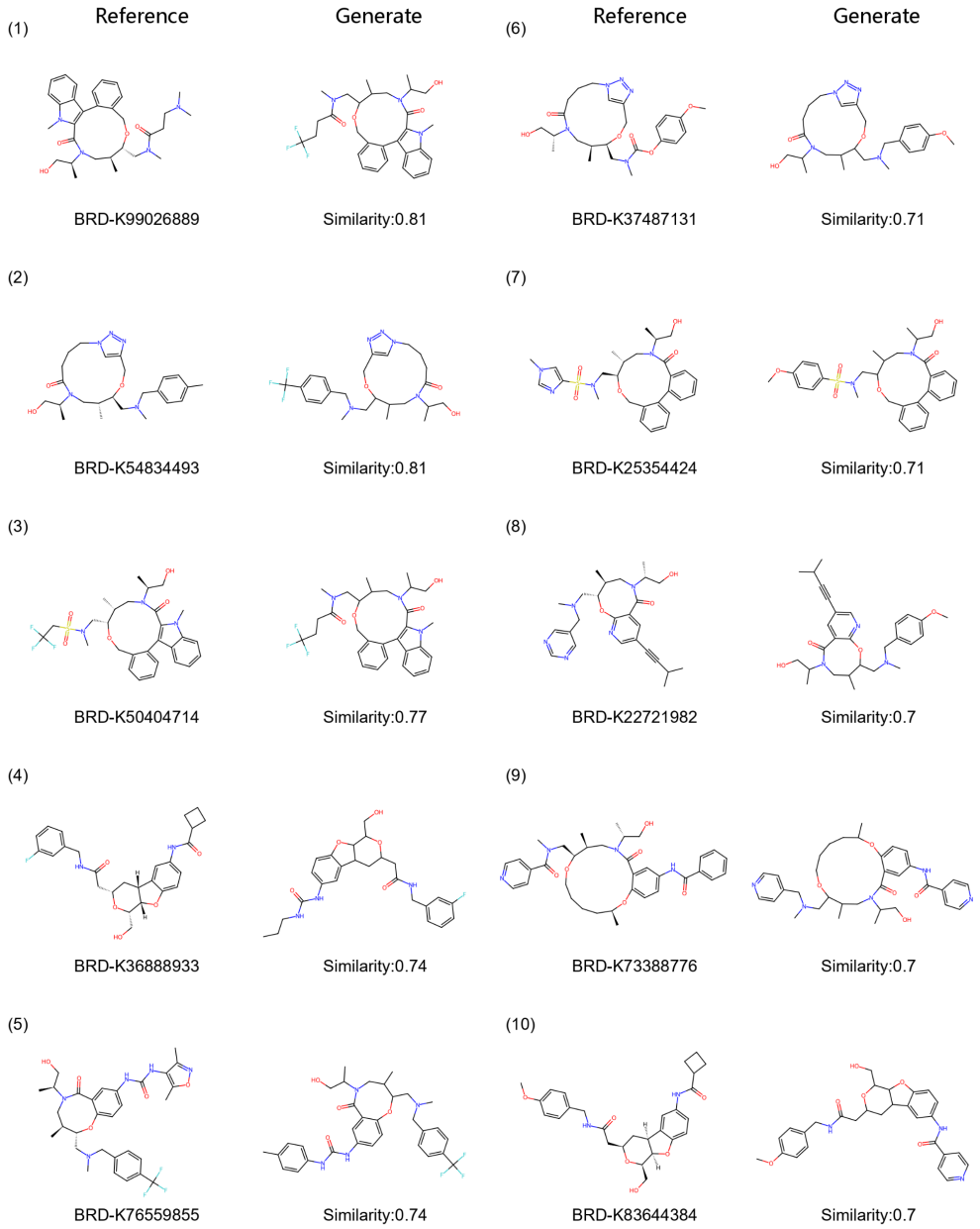


Fig. 2 This figure showcases examples of hit-like small molecules, generated by GexMolGen, from compound-induced gene expression signatures, compared to reference small molecules. The similarity calculation metric is based on the Tanimoto similarity of Morgan fingerprints.

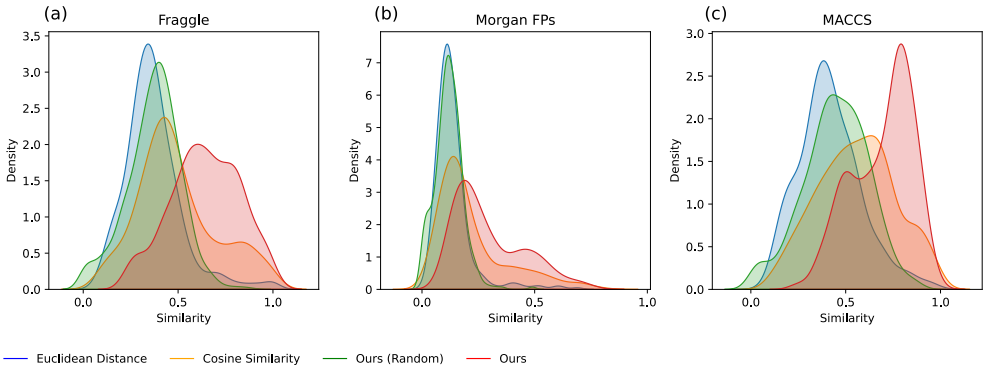


Fig. 3 The comparison results in the generation of hit-like molecules from compound-induced gene expression signatures. (a), (b), and (c) represent the distributions of metrics calculated using MACCS keys, Morgan fingerprints, and Fraggle similarity, respectively. The comparison methods include traditional searching methods such as cosine similarity and Euclidean distance, as well as the random model and ours, each represented by different colors.

2.3 Generating Hit-like Molecules from Knock-out Gene Expression Signatures

To test the generalization ability of our model, we challenge it to generate potential gene inhibitors from knock-out gene expression data, which can also be regarded as an out-of-distribution task. The underlying hypothesis is that inhibitors of a specific gene will cause equivalent changes in gene expression as when that gene is knocked out.

We utilize the same evaluation metrics and comparison methods as described in the Result 2.2. The collection of knock-out data and the corresponding known inhibitors can be found in the Methods 4.1.2. The procedure involves each method generating 100 candidate molecules for each gene based on the given knock-out transcriptome. Subsequently, we record the scores between each candidate and its most similar known inhibitors.

Fig.4 presents a visualization of the molecules generated by our model and the most similar known inhibitors. This similarity is measured by Morgan-based Tanimoto similarity scores. Compared to WGAN [4] with an average score of 0.26 (see Fig.B1 and Table A.5 for details), our model generates molecules that not only exhibit a higher overall degree of similarity to known inhibitors, but also successfully reconstructs the complete skeletal structure of the molecules (refer to Fig.4(4)(5)(6)).

The results of the experiments are fully demonstrated in Fig.5. Generally speaking, our model surpasses the three other methods with higher average scores and diversity, with the only exception being the Fraggle score in the task of identifying MTOR inhibitors. However, our model achieves the highest Fraggle score of 0.96 and consistently outperforms traditional searching methods in terms of MACCS-based and Morgan-based Tanimoto similarity scores. We speculate that this could be due to the fact that the training set contains many molecular substructures similar to known MTOR inhibitors, which results in higher Fraggle scores when directly searching the training set. Low scores in MACC and Morgan similarity highlights the need for comparative references to all the metrics. Furthermore, due to the limited size of the searching database, traditional methods' molecule candidates exhibit very low diversity. For instance, CTSK in MACCS-based metrics, shows the original box taking on a linear form. The random method, on the other hand, produces the widest range of scores due to the unconditional molecular sampling, but this also leads to molecular property deterioration. It occasionally produces molecules with high similarity scores, but it also generates a considerable number of molecules with very low correlations to known inhibitors, as observed in Fig.5's AKT2. By leveraging genetic information, our model narrows the generation space and allows for some diversity, striking a balance between property similarity and molecular diversity. We also refer to how other deep learning methods perform on similar tasks, as presented in Table 1. Despite some differences in the tested genes, it can be observed that our model consistently produces competitive results. The notable advantages of our model are its ability to generate molecules with 100% molecular validity and high input flexibility, features that are missing in other methods. The results from the experiments can be found in Table A.6, A.7, A.8.

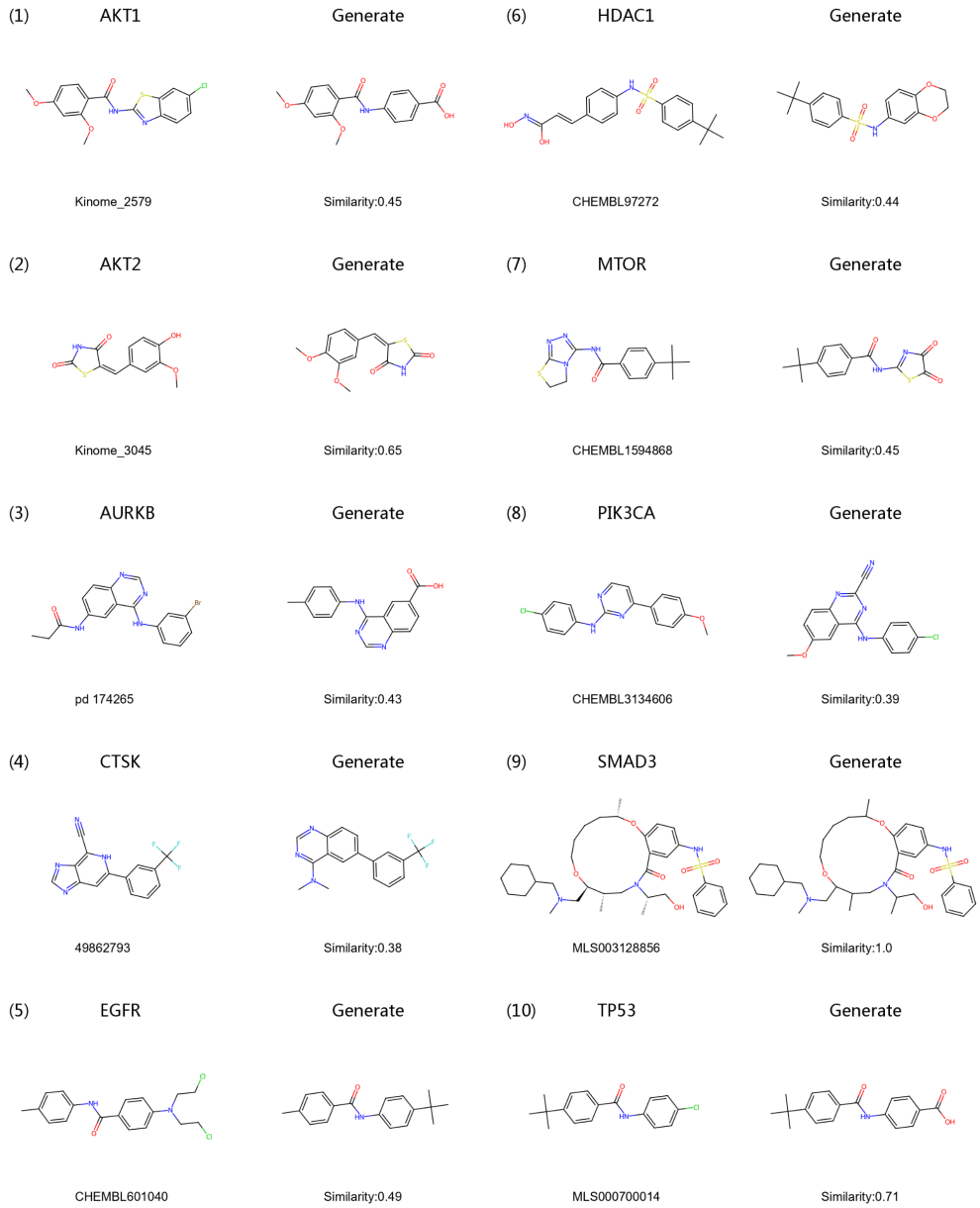


Fig. 4 This figure illustrates examples of potential gene inhibitors, generated by GexMolGen, using knock-out gene expression signatures. These are compared to known gene inhibitors, with the Tanimoto similarity of Morgan fingerprints as the calculation metric for similarity. For fair comparison, we have chosen the same 10 test genes as WGAN [4]. The results from WGAN can be viewed in Fig.B1.

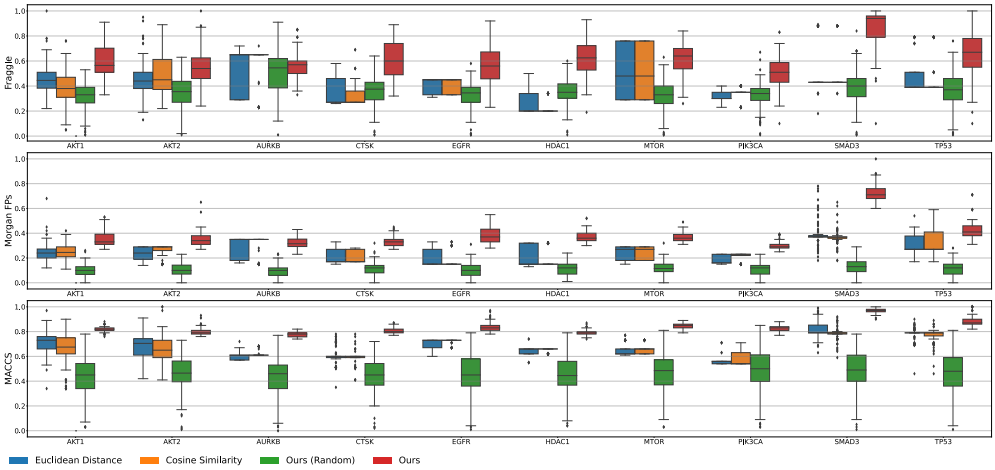


Fig. 5 This figure demonstrates the comparison results of potential gene inhibitors generated from knock-out gene expression signatures. The x-axis denotes three metrics: MACCS keys, Morgan fingerprints, and Fraggle similarity. The y-axis denotes the 10 test genes. Different methods are represented by different colors.

Method	Input Data Type	Tanimoto Similarity
WGAN [4]	L1000	$0.15 \sim 0.70$
Bidirectional Adversarial AE [26]	L1000	$0.30 \sim 0.48$
TRIOMPHE [27]	L1000	$0.29 \sim 0.68$
PaccMann [28]	GDSC/CCLE	$0.27 \sim 0.45$
Deep Generative with RL [29]	TCGA project	$0.47 \sim 0.83$
Ours	L1000	$\mathbf{0.74 \sim 1.00}$

Table 1 This is a comparison of GexMolGen with other deep learning approaches in generating potential inhibitors from knock-out transcriptomes. The first column outlines these methods, the second column specifies the types of datasets, and the third column outlines the range of Tanimoto similarity.

3 Discussion

In this paper, we propose GexMolGen, a method designed to generate hit-like molecules based on signatures calculated by desired gene expression relative to the initial states. We adopt a "first-align-then-generate" [20] strategy for cross-modal generation. To achieve this, we utilize an advanced single-cell large language model called scGPT [12] as the genetic encoder and pre-train hierVAE [12], a graph-based molecular model. To enhance the effects of contrastive learning, we integrate an auxiliary-guidance approach for alignment, serving as the mining task of hard negative samples. During the generation process, the facilitator initially transitions genetic embeddings into molecular tokens. These tokens are subsequently transformed into a continuous molecular embedding, which is then fed into the decoder to generate hit-like molecules. Users can increase diversity in generation by adjusting the beam width. This parameter corresponds to the candidate counts.

Our method demonstrates competitive and even superior performance compared to traditional search methods in generating molecules from both in-domain compound-induced expression signatures and out-of-domain gene knock-outs. Additionally, our model exhibits efficiency by ensuring 100% generation validity and supports input flexibility, surpassing other deep-learning methods. It also provides an explicit metric for measuring the similarity between genetic and molecular modalities.

Despite these advantages, our model may encounter specific limitations, such as the representation form of genes and the generation quality of existing molecular models. Existing single-cell foundation models represent genes in a sequence format, while gene interactions typically exist in a network form. Aggregating information from network neighbors using a graph neural network, based on predefined biological networks, may be a better approach to mimic biological activity. Furthermore, the appropriate molecular representation in generation is still remain an open question. SMILES-based models [30–32] have higher reconstruction accuracy but can be lossy due to reducing molecules to one-dimensional information. Graph-based models like MoFlow [18] and JT-VAE [33] show promise, but they can either be computationally expensive or difficult to achieve a satisfying reconstruction rate. Therefore, there is still much exploration needed to find a molecular generator that can fully leverage its strengths and mitigate its weaknesses.

In the future, we would like to apply GexMolGen in a real drug discovery setting to design molecules with specific biological effects and increase the chances of finding promising hits in the high-throughput screening (HTS).

4 Methods

4.1 Dataset Preprocessing

4.1.1 Dataset Used for Training

The datasets used in GexMolGen are derived from the L1000 public dataset (GEO ID: GSE70138) [34]. The L1000 technology uses microarray measurements to record the gene expression signatures caused by over 25,000 perturbations, including around

19,800 small molecules, in cell lines. This dataset is processed hierarchically and divided into 5 levels: Levels 1-3 record the absolute expression values of each gene in each bead, while Levels 4-5 report the differential expression of each gene obtained by comparing samples within each bead. During the data preprocessing stage, three issues are considered:

1. The pre-trained model scGPT [12] is trained on absolute gene expression values, while most related work [3–5] uses gene expression signatures in Level 5.
2. The sample sizes for each small molecule are extremely imbalanced, with the extreme ratio being as high as 22.
3. Splitting the dataset based on SMILES representations may result in the same molecule being included in both the training and hold-out sets, due to the non-uniqueness of SMILES.

Thus, we process the data as follows:

1. Use Level 3 data to encode both the perturbed and control groups, and then utilize the difference in the embedding layer to represent gene expression signatures.
2. Average gene expression to avoid bias from molecules with higher sample quantities
3. Partition the training, validation, and hold-out sets using the main layer of the InChI-key [35], thereby maximizing the difference among them.

Specifically, we first collect the perturbed and control data from Level 3, ensuring they originate from the same bead. This data belongs to 6 major cell lines – VCAP, PC3, A549, A375, HT29, and MCF7. The average gene expression is then calculated for each small molecule, resulting in a final dataset size of $2 \times N \times M$. Here, 2 signifies the experimental and control groups, N represent the sample size, and M indicates the count of gene symbols. Subsequently, we classify the structures based on the main layer of the InChI-key, resulting in a total of 6360 classes. From this set, we randomly select 300 structures, which include N_2 types of small molecules, to form the hold-out set. Additionally, we choose another 300 structures, including N_3 types of small molecules, to create the validation set. The remaining data is used for training the model, with N_1 types of small molecules.

For the molecular pre-training stage, we randomly sample N_p molecules from ChEMBL [25] as datasets.

4.1.2 Dataset Used for Evaluating

Besides the hold-out compound-induced gene expression data, we follow the same procedure as WGAN [4], our benchmarking method, to gather known inhibitors and genetic inputs for the comparative analysis of molecules generated from knock-out gene expressions. Initially, we select 10 known inhibitors of human genes from the ExCape dataset [36]. These genes are AKT1, AKT2, AURKB, CTSK, EGFR, HDAC1, MTOR, PIK3CA, SMAD3, and TP53. Our selection criteria consist of a Pic50 value greater than or equal to 5, non-presence of the same InChi-key in the training set, and compliance with valence bond rules. Eventually, we collect 2,153, 1,293, 2,122, 1,609, 4,244, 1,939, 2,858, 2,292, 23,758, and 12,860 known inhibitors, the numbers respectively corresponding to each gene.

For each of these 10 genes, we extract 100 transcriptome profiles from both the perturbed and control groups in the CRISPR knock-out perturbation data of the L1000 dataset [34].

4.2 Molecular Encoder

For the molecular encoder (see Fig.1C), hierVAE [19] regards motifs as the smallest building blocks for generating small molecules. This approach achieves 100% validity in molecular generation. It encodes the input molecule hierarchically to form an atom layer $\mathcal{H}_{\mathcal{G}}^{g}$, an attachment layer $\mathcal{H}_{\mathcal{G}}^{a}$, and a motif layer $\mathcal{H}_{\mathcal{G}}^{s}$. Each successive layer utilizes the previous one as its encoding condition.

Atom Layer.

The input for the atom layer $\mathcal{H}_{\mathcal{G}}^{g}$ is the molecular graph $\mathcal{G}$, the edge features $\{\mathrm{e}\left(\mathrm{b_{uv}}\right)\}$, and the node features $\{\mathrm{e}\left(\mathrm{a_u}\right)\}$. Through the message passing neural network [37, 38] $MPN_{\psi}(\cdot)$, we obtain the encoded features:

$$\mathrm{c}_{\mathcal{G}}^{\mathrm{g}} = \{\mathrm{h_v}\} = MPN_{\psi_1}\left(\mathcal{H}_{\mathcal{G}}^{g}, \{\mathrm{e}\left(\mathrm{a_u}\right)\}, \{\mathrm{e}\left(\mathrm{b_{uv}}\right)\}\right), \tag{1}$$

where $\mathrm{h_v}$ represents the updated atomic features.

Attachment Layer.

The attachment layer $\mathcal{H}_{\mathcal{G}}^{a}$ updates its nodes using the following equation:

$$\mathrm{c}_{\mathcal{G}}^{a} = \{\mathrm{h}_{\mathcal{A}_\mathrm{i}}\} = MPN_{\psi_2}\left(\mathcal{H}_{\mathcal{G}}^{g}, \{f_{\mathcal{A}_\mathrm{i}}\}, \{\mathrm{e}\left(\mathrm{d_{ij}}\right)\}\right), \tag{2}$$

where $\mathrm{e}\left(\mathrm{d_{ij}}\right)$ denotes the edge features for the attachment layer, and $f_{\mathcal{A}_i}$ can be calculated based on atom vectors $\mathrm{h_v}$ and nodes $\mathcal{A}_i$ in the attachment layer:

$$f_{\mathcal{A}_\mathrm{i}} = emb_{\mathcal{A}_\mathrm{i}}\left(\mathrm{e}\left(\mathcal{A}_\mathrm{i}\right), \sum_{\mathrm{v}\in\mathrm{S_i}} \mathrm{h_v}\right), \tag{3}$$

where $emb_{\mathcal{A}_i}(\cdot)$ represents a fully connected layer and $e(\cdot)$ symbolizes a conventional embedding layer. The node $\mathcal{A}_i$ is defined as the set of overlapping atoms between adjacent motifs.

Motif Layer.

The motif layer $\mathcal{H}_{\mathcal{G}}^{s}$ updates its nodes using the following equation:

$$\mathrm{c}_{\mathcal{G}}^{s} = \{\mathrm{h}_{\mathcal{S}_\mathrm{i}}\} = MPN_{\psi_3}\left(\mathcal{H}_{\mathcal{G}}^{s}, \{f_{\mathcal{S}_\mathrm{i}}\}, \{\mathrm{e}\left(\mathrm{d_{ij}}\right)\}\right), \tag{4}$$

where $\mathrm{e}\left(\mathrm{d_{ij}}\right)$ denotes the edge features for the motif layer, and $f_{\mathcal{S}_i}$ can be calculated based on $\mathrm{h}_{\mathcal{A}_i}$ and $\mathcal{S}_i$:

$$f_{\mathcal{S}_\mathrm{i}} = emb_{\mathcal{S}_\mathrm{i}}\left(\mathrm{e}\left(\mathcal{S}_\mathrm{i}\right), \mathrm{h}_{\mathcal{A}_\mathrm{i}}\right), \tag{5}$$

where $emb_{\mathcal{A}_i}(\cdot)$ represents a fully connected layer and $e(\cdot)$ symbolizes a conventional embedding layer. Nodes $\mathcal{S}_i$ in the motif layer represent the type of the motifs.

Finally, the embedding of the i-th molecule is defined as:

$$z_m^{(\mathrm{i})} = \mu\left(\mathrm{h}_{\mathcal{S}_1}\right) + \exp\left(\sum\left(\mathrm{h}_{\mathcal{S}_1}\right)\right) \cdot \epsilon, \epsilon \sim \mathcal{N}(0, \mathrm{I}), \tag{6}$$

where $\mu(\cdot)$, $\exp(\cdot)$ are both fully connected layers representing the mean and log variance, respectively, and $\mathcal{S}_1$ represents the root motif, which is also the first motif for decoding.

4.3 Molecular Decoder

During decoding (see Fig.1E), we continue to adopt the molecular embedding notion as z_m, and the motif layer encoded vectors as $h_{\mathcal{S}_k}$, where k signifies the step. Initially, we set $k = 1$. Then, the hierarchical decoding of the molecular graph is shown as follows.

Motif Prediction.

This step predicts the next motif $\mathcal{S}_t$ to be attached to $\mathcal{S}_k$. It is implemented as a classification task over the motif vocabulary $\mathcal{V}_{\mathcal{S}}$:

$$p_{\mathcal{S}_t} = \mathrm{softmax}(\mathrm{MLP}(h_{\mathcal{S}_k}, z_m)). \tag{7}$$

Attachment Prediction.

Similar to motif prediction, but applying to the attachment vocabulary $\mathcal{V}_{\mathcal{A}}(\mathcal{S}_t)$, this step predicts the attachment configuration between $\mathcal{S}_t$ and $\mathcal{S}_k$. In other words, it determines the type of atom that connects $\mathcal{S}_t$ and $\mathcal{S}_k$:

$$p_{\mathcal{A}_\iota} = \mathrm{softmax}(\mathrm{MLP}(h_{\mathcal{S}_k}, z_m)). \tag{8}$$

Graph Prediction.

This step is to decide how $\mathcal{S}_t$ will be attached to $\mathcal{S}_k$. After attachment prediction, we only narrow down to the connected atom type, but we don't specify the exact position. The attachment between $\mathcal{S}_t$ and $\mathcal{S}_k$ is defined as atom pairs $\mathcal{M}_{tk} = \{(u_j.v_j)|u_j \in \mathcal{A}_k, v_j \in \mathcal{A}_t\}$ where atom u_j and v_j are attached together. The probability of a candidate attachment, M, is calculated based on the atom vectors, h_{u_j} and h_{v_j}:

$$p_M = \mathrm{softmax}\left(h_M \cdot z_m\right), \tag{9}$$

$$h_M = \textstyle\sum_j \mathrm{MLP}(h_{u_j}, h_{v_j}). \tag{10}$$

Training.

During training, this module needs to minimize the negative ELBO:

$$\mathcal{L}_{\mathrm{ELBO}} = -\mathbb{E}_{z_m \sim Q}[\log P(\mathcal{G}|z_m)] + \lambda_{KL}\mathcal{D}_{KL}[Q(z_m|\mathcal{G})||P(z_m)] \tag{11}$$

to complete the training loop.

Validation.

During validation, we utilize the Fréchet ChemNet Distance (FCD) [39] as a selection criterion for models. This metric evaluates whether the generated molecules show diversity and possess chemical and biological properties similar to those of real molecules. The training details can be found in Fig.B2.

4.4 Genetic Encoder

In Fig.1B, a single-cell large language model known as scGPT [12] is employed as the genetic encoder. Trained on an extensive amount of unlabeled single-cell data, this model show potential in capturing underlying biological signals. The pretrained gene tokens from scGPT identities each gene in the input data, allowing for flexible gene inputs during the encoding process. Our model is designed to receive two sets of gene expression data: the initial and the desired or post-perturbation data. Both sets of data independently undergo the same encoding pipeline. Using the i-th gene expression data point as an example, we initially retrieve its corresponding gene tokens from the scGPT pretrained gene dictionary:

$$t_g^{(i)} = \left[id\left(g_1^{(i)}\right), id\left(g_2^{(i)}\right), \ldots\ldots, id\left(g_M^{(i)}\right)\right], \tag{12}$$

where $t_g^{(i)}$ represents the gene token encoding for the i-th data, and $id\left(g_1^{(j)}\right)$ represents the unique integer identifier for gene g_j in the i-th data.

Next, we preprocess the expression values by implementing transcripts per thousand (TPT) normalization and $log1p$ transformation:

$$x^{(i)} = \left[x_1^{(i)}, x_2^{(i)} \ldots\ldots x_M^{(i)}\right]. \tag{13}$$

The gene embedding is then defined as follows:

$$z_g^{(i)} = emb_g\left(t_g^{(i)}\right) + emb_x\left(x^{(i)}\right), \tag{14}$$

where $emb_g(\cdot)$ refers to the conventional embedding layers, and $emb_x(\cdot)$ represents the fully connected layer.

We utilize the mask language model (MLM) task to further fine-tune scGPT. Instead of commonly using mean square loss (MSE), we turn to the autofocus direction-aware loss in GEARS [40]. This loss automatically assigns a greater weight to differentially expressed genes by amplifying the exponent of the error. Given a mini-batch consisting of B_1 data points, we randomly sample 95% of the genes and mask 10% of them. The genetic encoder then predicts the expression of these masked genes, which is denoted as $\hat{g}$, and compared it with their actual expression g. The loss is

defined as follows:

$$\mathcal{L}_{\text{autofocus}} = \frac{1}{B_1} \sum_{l=1}^{B_1} \frac{1}{M_1} \sum_{j=1}^{M_1} (g_j - \hat{g}_j)^{(2+\gamma)} . \tag{15}$$

Moreover, an additional direction-aware loss is introduced to increase predicted data's sensitivity to the direction relative to the control group. This is defined as follows:

$$\mathcal{L}_{\text{direction}} = \frac{1}{B_1} \sum_{l=1}^{B_1} \frac{1}{M_1} \sum_{j=1}^{M_1} \left[\text{sign}\left(g_j - g_j^{\text{ctrl}}\right) - \text{sign}\left(\hat{g}_u - g_j^{\text{ctrl}}\right)\right]^2 . \tag{16}$$

The optimization objective for the MLM task, denoted as $\mathcal{L}_{\text{MLM}}$, is ultimately calculated by taking a weighted sum of the values of $\mathcal{L}_{\text{autofocus}}$ and $\mathcal{L}_{\text{direction}}$:

$$\mathcal{L}_{\text{MLM}} = \mathcal{L}_{\text{autofocus}} + \lambda \mathcal{L}_{\text{direction}}, \tag{17}$$

where λ functions to adjust the weight for the directionality loss. The training details can be visualized in Fig.B4.

We then represent the gene expression signatures for subsequent alignment by calculating the difference in post encoding embedding between the desired and control states:

$$z_g = z_g^{\text{desired}} - z_g^{\text{control}}. \tag{18}$$

4.5 Alignment

Alignment is the 2nd step (see Figure.1A) and aims to extract the mutual information between genetic and small molecular modalities, thus enhancing the controllability of the generation model. In this process, we adopt contrastive learning [41, 42] for multi-modal matching and introduce an auxiliary-guidance strategy to overcome model collapse.

Model collapse occurs when different input vectors yield nearly identical output vectors. We hypothesize that this is due to the model struggling to discern positive and negative sample pairs in the early epochs of training, resulting in weak gradient signals that hinder the model's learning process. Additionally, we observe two phenomena: one is that if we use cross-entropy loss, the model does not collapse, but due to the large number of small molecule types, the model does not converge at a satisfying result. Another one is the genetic encoder being more prone to collapse. Based on our analysis and observations, we take the following steps to assist the learning of the Facilitator [41].

4.5.1 Hierarchical clustering.

Hierarchical clustering merges data into clusters step by step by calculating the similarity between them, until a complete tree-like structure is formed. We use this strategy to cluster the small molecule embeddings into $C = \{c_1, c_2, c_3\}$ classes. The detailed process is shown in Algorithm 1.

Algorithm 1 Hierarchical Clustering

Input: Small molecule embedding vectors $Z_M = \{z_{m_1}, z_{m_2}, \ldots, z_{m_{N1}}\}$
    Distance function $\texttt{dist}(c_1, c_2)$
Output: Categories C of small molecule embedding vectors after hierarchical clustering at each level.
for $i = 1$ to N_1 **do**
    $c_i = \{Z_{m_i}\}$
end for
$C = \{c_1, c_2, \ldots, c_{N_1}\}$
$I = n + 1$
while $C.\texttt{size} > 10$ **do**
    $(c_{min_1}, c_{min_2}) = \min \texttt{dist}(c_i, c_j)$ for all $c_i, c_j \in C$
    $C = C - \{c_{min_1}, c_{min_2}\}$
    $C = C \cup \{(c_{min_1}, c_{min_2})\}$
    $I = I + 1$
end while
return C

4.5.2 Auxiliary-guided gene encoding.

In the genetic encoder, the output of the final layer z_g is aligned with small molecular vectors z_m via contrastive learning. The optimization objective is to minimize the InfoNCE loss function [43]:

$$\begin{aligned}\mathcal{L}_{\text{InfoNCE}} = &-\frac{1}{2}E_{z_m z_g}\left[log\frac{exp\left(E(z_m z_g)\right)}{exp\left(E(z_m z_g)\right)+\sum_{g'} exp\left(E(z_m, z_g')\right)}\right] \\ &-\frac{1}{2}E_{z_m z_g}\left[log\frac{exp\left(E(z_m z_g)\right)}{exp\left(E(z_m z_g)\right)+\sum_{m'} exp\left(E(z_m' z_g)\right)}\right]\end{aligned}, \tag{19}$$

where ${z_m}'$ and ${z_g}'$ are unpaired molecule-gene negative sample pairs in a minibatch composed of B_2 data points, $E(.,.)$ is an energy (scalar) function defined on these two modalities, and we use vector dot product for representation. The InfoNCE loss aims to minimize the distance between positive samples and maximize the distance between negative samples.

However, contrastive learning often faces model collapse. Gradient analysis [44] reveals that the gradients related to negative samples are proportional to their similarity to the anchor, proving that InfoNCE loss is a hardness-aware function. Therefore, exploring difficult negative samples is crucial for the final effect of contrastive learning. Unfortunately, in the field of life sciences, the importance of hard negative samples is often impractical due to the lack of ground truth relationships between different modalities. GexMolGen treats molecules that exhibit high similarity, but have distinct effects on gene expression, as hard negative samples. Through hierarchical clustering, molecules are grouped into clusters, and the model can be guided by the cross-entropy loss function at a coarse resolution. Given that the small molecules have been clustered into $C = \{c_1, c_2, c_3\}$ classes, we require the genetic encoder to classify the corresponding perturbations based on gene expression signatures. The quality of this task is measured by the cross-entropy loss function:

$$\mathcal{L}_{\text{CE}_\text{c}} = -\sum_{i=1}^{N_1} y_{c,i} log(p_{c,i}), \tag{20}$$

where $y_{c,i}$ and $p_{c,i}$ denote the truth labels and predicted labels, respectively, for the c-th cluster of small molecules based on gene embeddings.

During the learning process, our gene encoder optimizes two functions: InfoNCE loss $\mathcal{L}_{\text{InfoNCE}}$ for contrastive learning, and cross-entropy loss $\mathcal{L}_{\text{CE}_\text{c}}$ for the auxiliary guidance serving as the hard sample mining task:

$$\mathcal{L}_{\text{contrast}} = \mathcal{L}_{\text{InfoNCE}} + \sum_{i=1}^{C} \mathcal{L}_{\text{CE}_\text{c}}. \tag{21}$$

The details of the training process can be seen in Fig.B4.

4.6 Facilitator

Assuming our dataset consists of sample pairs (m, g) where m represents a small molecule and g represents its corresponding gene expression signatures, we model the transition from the gene expression signatures to the molecules, denoted as $P(m|g)$, using a two-phase process involving modality facilitation and model generation [45]. Let z_m and z_g continue to represent the embeddings of the small molecule and gene, respectively. Thus, $P(m|g)$ can be decomposed as

$$P(m|g) = P(m, z_m|g) = P(m|z_m, g)P(z_m|g). \tag{22}$$

The first equation holds because z_m is a deterministic function of the small molecule dataset m, and the second equation holds due to the chain rule [20, 21].

When the small molecule decoder $P(m|z_m, g)$ has been determined [19], the primary focus is on optimizing the facilitator $P(z_m|g)$ (see Fig.1D). We used simple multi-layer perceptrons (MLPs) as this module, but the results were unsatisfactory, with a loss that was almost 30% higher. Additionally, it did not help further exploration of the relationship between molecular substructure and specific gene differential expression. Taking these limitations into account, we adopt an auto-regressive model for modality transformation [46–48].

When selecting the loss function for the auto-regressive model, cross-entropy is often more suitable than mean squared error (MSE). Since the former is well-suited for handling probability distributions, and its derivative directly captures the differences between probability distributions. This facilitates gradient computation and allows for the observation of significant changes during training. However, since z_m is a continuous value, the cross-entropy loss function cannot be directly applied. To solve this, we adopt the approach utilized in DALL·E 2 [21], where we reduce the dimensionality of z_m from d to L using principal component analysis (PCA). This reduction allows us to retain nearly all the information (with a reconstruction mean squared error below 1%). Subsequently, we discretize the L-dimensional values into N_m bins, assuming that these values are uniformly distributed.

Fig.1D provides the mechanism of the Facilitator. It takes as input M genes, a special symbol [start], and L small molecule tokens, and sequentially produces inferred tokens z_m^d. The loss function is defined as follows:

$$\mathcal{L}_{\text{facilitator}} = E_{z_m^d}[-\sum_{i=1}^{L} t_i log(p(z_m^{d(i)}|z_m^{d(0:i-1)}))], \tag{23}$$

where t_i represents the i-th true token, and $p(z_m^{d(i)}|z_m^{d(0:i-1)})$ represents the inferred token of the small molecules.

Beam Search Optimization.

Beam search is a commonly used search algorithm in sequence generation tasks [16, 49]. It utilizes the principles of dynamic programming by keeping track of a set of candidate sequences, known as the beam, with a beam width of K to preserve promising results.

The beam search algorithm effectively manages the search space and improves the accuracy and diversity of generated sequences. Specifically, the beam search algorithm follows the following steps:

1. Initialize the beam with a set of candidate sequences, each consisting of M gene tokens and [start].
2. At each time step:
   (a) Calculate the likelihood of every candidate token becoming the next token based on the current sequence.
   (b) Select the top K candidate sequences based on their scores, where K is the beam width.
   (c) Expand each selected candidate sequence by appending the molecular tokens and update its score.
3. Repeat steps 2 until a complete sequence of small molecule tokens L is formed.
4. Return the top-ranked sequence from the beam as the final output.

After the beam search process, the final output tokens z_m^d are transformed into z_m using the inverse of PCA. The continuous z_m values are then fed into the molecular decoder $P(m|z_m, g)$ [19]. This decoding procedure results in K sets of molecule candidates.

Declarations

Funding

This work was supported by grants from National Natural Science Foundation of China 62103262 (to Y.Y.) and the Shanghai Pujiang Programme (no. 21PJ1407700 to Y.Y.).

Author contributions

The concept of this work was conceived by J.C., Q.Y. and Y.Y. The experiments were performed by J.C., with the assistance of X.P., K.Y., S.C. and B.L. The results were analyzed by J.C. All authors participated in the writing of the manuscript. The project was under the supervision of X.P., Q.Y. and Y.Y.

Competing financial interests

The authors declare no competing financial interests.

References

[1] Vincent, F. *et al.* Phenotypic drug discovery: recent successes, lessons learned and new directions. *Nat Rev Drug Discov* **21**, 899–914 (2022). URL https://www.nature.com/articles/s41573-022-00472-w. Number: 12 Publisher: Nature Publishing Group.

[2] Musa, A. *et al.* A review of connectivity map and computational approaches in pharmacogenomics. *Briefings in Bioinformatics* **18**, 903 (2017). URL https://doi.org/10.1093/bib/bbx023.

[3] Zhu, J. *et al.* Prediction of drug efficacy from transcriptional profiles with deep learning. *Nat Biotechnol* **39**, 1444–1452 (2021). URL https://www.nature.com/articles/s41587-021-00946-z. Number: 11 Publisher: Nature Publishing Group.

[4] Méndez-Lucio, O., Baillif, B., Clevert, D.-A., Rouquié, D. & Wichard, J. De novo generation of hit-like molecules from gene expression signatures using artificial intelligence. *Nat Commun* **11**, 1–10 (2020). URL https://www.nature.com/articles/s41467-019-13807-w. Number: 1 Publisher: Nature Publishing Group.

[5] Das, D., Chakrabarty, B., Srinivasan, R. & Roy, A. Gex2SGen: Designing Drug-like Molecules from Desired Gene Expression Signatures. *J. Chem. Inf. Model.* **63**, 1882–1893 (2023). URL https://doi.org/10.1021/acs.jcim.2c01301. Publisher: American Chemical Society.

[6] Testing the limits of SMILES-based de novo molecular generation with curriculum and deep reinforcement learning | Nature Machine Intelligence. URL https://www.nature.com/articles/s42256-023-00636-2#Sec2.

[7] Goodfellow, I. J. *et al.* Generative Adversarial Networks (2014). URL https://arxiv.org/abs/1406.2661v1.

[8] Boiarsky, R., Singh, N., Buendia, A., Getz, G. & Sontag, D. A Deep Dive into Single-Cell RNA Sequencing Foundation Models. *bioRxiv* 2023.10.19.563100 (2023). URL https://www.biorxiv.org/content/10.1101/2023.10.19.563100v1. Publisher: Cold Spring Harbor Laboratory Section: New Results.

[9] Yang, F. *et al.* scBERT as a large-scale pretrained deep language model for cell type annotation of single-cell RNA-seq data. *Nat Mach Intell* **4**, 852–866 (2022). URL https://www.nature.com/articles/s42256-022-00534-z. Number: 10 Publisher: Nature Publishing Group.

[10] Theodoris, C. V. *et al.* Transfer learning enables predictions in network biology. *Nature* **618**, 616–624 (2023). URL https://www.nature.com/articles/s41586-023-06139-9. Number: 7965 Publisher: Nature Publishing Group.

[11] Chen, J. *et al.* Transformer for one stop interpretable cell type annotation. *Nat Commun* **14**, 1–14 (2023). URL https://www.nature.com/articles/s41467-023-35923-4. Number: 1 Publisher: Nature Publishing Group.

[12] Cui, H. *et al.* scGPT: Towards Building a Foundation Model for Single-Cell Multi-omics Using Generative AI (2023). URL https://www.biorxiv.org/content/10.1101/2023.04.30.538439v2. Pages: 2023.04.30.538439 Section: New Results.

[13] Hao, M. *et al.* Large Scale Foundation Model on Single-cell Transcriptomics (2023). URL https://www.biorxiv.org/content/10.1101/2023.05.29.542705v4. Pages: 2023.05.29.542705 Section: New Results.

[14] Heimberg, G. *et al.* Scalable querying of human cell atlases via a foundational model reveals commonalities across fibrosis-associated macrophages (2023). URL https://www.biorxiv.org/content/10.1101/2023.07.18.549537v1. Pages: 2023.07.18.549537 Section: New Results.

[15] Yang, X. *et al.* GeneCompass: Deciphering Universal Gene Regulatory Mechanisms with Knowledge-Informed Cross-Species Foundation Model (2023). URL https://www.biorxiv.org/content/10.1101/2023.09.26.559542v1. Pages: 2023.09.26.559542 Section: New Results.

[16] Bresson, X. & Laurent, T. A Two-Step Graph Convolutional Decoder for Molecule Generation (2019). URL http://arxiv.org/abs/1906.03412. ArXiv:1906.03412 [cs, stat] version: 2.

[17] De Cao, N. & Kipf, T. MolGAN: An implicit generative model for small molecular graphs (2018). URL https://arxiv.org/abs/1805.11973v2.

[18] Zang, C. & Wang, F. *MoFlow: An Invertible Flow Model for Generating Molecular Graphs*, KDD '20, 617–626 (Association for Computing Machinery, New York, NY, USA, 2020). URL https://dl.acm.org/doi/10.1145/3394486.3403104.

[19] Jin, W., Barzilay, R. & Jaakkola, T. Hierarchical Generation of Molecular Graphs using Structural Motifs (2020). URL http://arxiv.org/abs/2002.03230. ArXiv:2002.03230 [cs, stat].

[20] Ramesh, A. *et al.* Zero-Shot Text-to-Image Generation (2021). URL http://arxiv.org/abs/2102.12092. ArXiv:2102.12092 [cs].

[21] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C. & Chen, M. Hierarchical Text-Conditional Image Generation with CLIP Latents (2022). URL http://arxiv.org/abs/2204.06125. ArXiv:2204.06125 [cs].

[22] Edwards, C., Zhai, C. & Ji, H. *Text2Mol: Cross-Modal Molecule Retrieval with Natural Language Queries*, 595–607 (Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021). URL https://aclanthology.org/2021.emnlp-main.47.

[23] Liu, S. *et al.* A Text-guided Protein Design Framework (2023). URL http://arxiv.org/abs/2302.04611. ArXiv:2302.04611 [cs, q-bio, stat].

[24] Huang, Z., Bianchi, F., Yuksekgonul, M., Montine, T. J. & Zou, J. A visual–language foundation model for pathology image analysis using medical Twitter. *Nat Med* **29**, 2307–2316 (2023). URL https://www.nature.com/articles/

s41591-023-02504-3. Number: 9 Publisher: Nature Publishing Group.

[25] Gaulton, A. *et al.* ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Res* **40**, D1100–D1107 (2012). URL https://dx.doi.org/10.1093/nar/gkr777. Publisher: Oxford Academic.

[26] Shayakhmetov, R. *et al.* Molecular Generation for Desired Transcriptome Changes With Adversarial Autoencoders. *Frontiers in Pharmacology* **11** (2020). URL https://www.frontiersin.org/articles/10.3389/fphar.2020.00269.

[27] TRIOMPHE: Transcriptome-Based Inference and Generation of Molecules with Desired Phenotypes by Machine Learning. URL https://pubs.acs.org/doi/epdf/10.1021/acs.jcim.1c00967.

[28] Born, J., Manica, M., Oskooei, A., Cadow, J. & Rodríguez Martínez, M. Schwartz, R. (ed.) *PaccMannRL: Designing Anticancer Drugs From Transcriptomic Data via Reinforcement Learning.* (ed.Schwartz, R.) *Research in Computational Molecular Biology*, Lecture Notes in Computer Science, 231–233 (Springer International Publishing, Cham, 2020).

[29] Pereira, T. *et al.* Deep generative model for therapeutic targets using transcriptomic disease-associated data—USP7 case study. *Brief Bioinform* **23** (2022). URL https://dx.doi.org/10.1093/bib/bbac270. Publisher: Oxford Academic.

[30] Blaschke, T. *et al.* REINVENT 2.0: An AI Tool for De Novo Drug Design. *J. Chem. Inf. Model.* **60**, 5918–5922 (2020). URL https://doi.org/10.1021/acs.jcim.0c00915. Publisher: American Chemical Society.

[31] Winter, R., Montanari, F., Noé, F. & Clevert, D.-A. Learning continuous and data-driven molecular descriptors by translating equivalent chemical representations. *Chemical Science* **10**, 1692–1701 (2019). URL https://pubs.rsc.org/en/content/articlelanding/2019/sc/c8sc04175j. Publisher: Royal Society of Chemistry.

[32] Segler, M. H. S., Kogej, T., Tyrchan, C. & Waller, M. P. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.* **4**, 120–131 (2018). URL https://pubs.acs.org/doi/10.1021/acscentsci.7b00512.

[33] Jin, W., Barzilay, R. & Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation (2018). URL https://arxiv.org/abs/1802.04364v4.

[34] Subramanian, A. *et al.* A Next Generation Connectivity Map: L1000 Platform and the First 1,000,000 Profiles. *Cell* **171**, 1437–1452.e17 (2017). URL https://www.cell.com/cell/abstract/S0092-8674(17)31309-0. Publisher: Elsevier.

[35] Heller, S., McNaught, A., Stein, S., Tchekhovskoi, D. & Pletnev, I. InChI - the worldwide chemical structure identifier standard. *J Cheminform* **5**, 1–9 (2013). URL https://jcheminf.biomedcentral.com/articles/10.1186/1758-2946-5-7. Number: 1 Publisher: BioMed Central.

[36] Sun, J. *et al.* ExCAPE-DB: an integrated large scale dataset facilitating Big Data analysis in chemogenomics. *Journal of Cheminformatics* **9**, 17 (2017). URL https://doi.org/10.1186/s13321-017-0203-5.

[37] Dai, H., Dai, B. & Song, L. Discriminative Embeddings of Latent Variable Models for Structured Data (2020). URL http://arxiv.org/abs/1603.05629. ArXiv:1603.05629 [cs].

[38] Fey, M., Yuen, J.-G. & Weichert, F. Hierarchical Inter-Message Passing for Learning on Molecular Graphs (2020). URL http://arxiv.org/abs/2006.12179. ArXiv:2006.12179 [cs, stat].

[39] Preuer, K., Renz, P., Unterthiner, T., Hochreiter, S. & Klambauer, G. Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery. *J. Chem. Inf. Model.* **58**, 1736–1741 (2018). URL https://doi.org/10.1021/acs.jcim.8b00234. Publisher: American Chemical Society.

[40] Roohani, Y., Huang, K. & Leskovec, J. Predicting transcriptional outcomes of novel multigene perturbations with GEARS. *Nat Biotechnol* 1–9 (2023). URL https://www.nature.com/articles/s41587-023-01905-6. Publisher: Nature Publishing Group.

[41] Khosla, P. *et al.* Supervised Contrastive Learning (2021). URL http://arxiv.org/abs/2004.11362. ArXiv:2004.11362 [cs, stat].

[42] Tian, Y., Krishnan, D. & Isola, P. *Contrastive Multiview Coding*, 776–794 (Springer-Verlag, Berlin, Heidelberg, 2020). URL https://doi.org/10.1007/978-3-030-58621-8_45.

[43] Oord, A. v. d., Li, Y. & Vinyals, O. Representation Learning with Contrastive Predictive Coding (2018). URL https://arxiv.org/abs/1807.03748v2.

[44] Wang, F. & Liu, H. Understanding the Behaviour of Contrastive Loss (2021). URL http://arxiv.org/abs/2012.09740. ArXiv:2012.09740 [cs].

[45] Ding, M. *et al.* CogView: Mastering Text-to-Image Generation via Transformers (2021). URL http://arxiv.org/abs/2105.13290. ArXiv:2105.13290 [cs].

[46] Radford, A. *et al.* Language Models are Unsupervised Multitask Learners .

[47] Lewis, M. *et al.* BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension (2019). URL http://

arxiv.org/abs/1910.13461. ArXiv:1910.13461 [cs, stat].

[48] Raffel, C. *et al.* Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (2020). URL http://arxiv.org/abs/1910.10683. ArXiv:1910.10683 [cs, stat].

[49] OpenAI. GPT-4 Technical Report (2023). URL http://arxiv.org/abs/2303.08774. ArXiv:2303.08774 [cs].

[50] Fey, M. & Lenssen, J. E. Fast Graph Representation Learning with PyTorch Geometric (2019). URL https://arxiv.org/abs/1903.02428v3.

[51] Landrum, G. RDKit: Open-source cheminformatics. Release 2014.03.1 (2014). URL https://zenodo.org/records/10398.

[52] Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library (2019). URL https://arxiv.org/abs/1912.01703v1.

Appendix A Supplementary Tables

A.1 Settings

The experiments run on a machine that is equipped with two Intel Xeon Gold 6138 CPUs, four RTX 3090 GPUs, and 503GB of RAM. We utilize the PyG library [50] for constructing graphs and RDKit [51] for preprocessing molecules. The remaining components are implemented in PyTorch [52].

Symbol	Description	Value
N	The size of the dataset.	15156
N_1	The size of the training dataset.	13757
N_2	The size of the valid dataset.	708
N_3	The size of the hold-out dataset.	691
N_m	The number of discrete molecular bins.	1024
N_p	The size of the molecular pre-training dataset.	128369
D	The used depth of pre-trained scGPT [12].	8
M	The count of gene symbols.	978
M_1	The count of masked gene symbols per minibatch.	93
B_1	Batch size for steps 1, 3, and 4.	32
B_2	Batch size for steps 2.	256
γ	The hyper-parameter associated with the autofocus loss $\mathcal{L}_{\text{autofocus}}$.	2
λ	The weight assigned to the direction-aware loss $\mathcal{L}_{\text{direction}}$.	0.001
d	The dimensionality of alignment encodings.	64
L	The dimensionality of molecular encodings after PCA (Principal Component Analysis).	43
c_1	The first hierarchical category.	1000
c_2	The second hierarchical category.	100
c_3	The third hierarchical category.	10
K	The beam width.	20

Table A1 The hyper-parameters and their corresponding values in GexMolGen.

A.2 Supplementary Table 1

Experimental results from four methods that generate hit-like molecules, using a Tanimoto similarity metric based on Morgan fingerprints in the compound-induced gene expression.

A.3 Supplementary Table 2

Experimental results from four methods that generate hit-like molecules, using a Tanimoto similarity metric based on MACCS keys in the compound-induced gene expression.

A.4 Supplementary Table 3

Experimental results from four methods that generate hit-like molecules, using the Fraggle similarity metric in the compound-induced gene expression.

A.5 Supplementary Table 4

The SMILES of generated molecules and their most similar known inhibitors from the WGAN [4]. The molecules were obtained using optical character recognition (OCR). The metric used to measure similarity is the Tanimoto coefficient, based on Morgan fingerprints.

A.6 Supplementary Table 5

Experimental results from four methods that generate potential gene inhibitors, using a Tanimoto similarity metric based on Morgan fingerprints in the knock-out gene expression.

A.7 Supplementary Table 6

Experimental results from four methods that generate potential gene inhibitors, using a Tanimoto similarity metric based on MACCS keys in the knock-out gene expression.

A.8 Supplementary Table 7

Experimental results from four methods that generate potential gene inhibitors, using the Fraggle similarity metric in the knock-out gene expression.

Appendix B Supplementary Figures

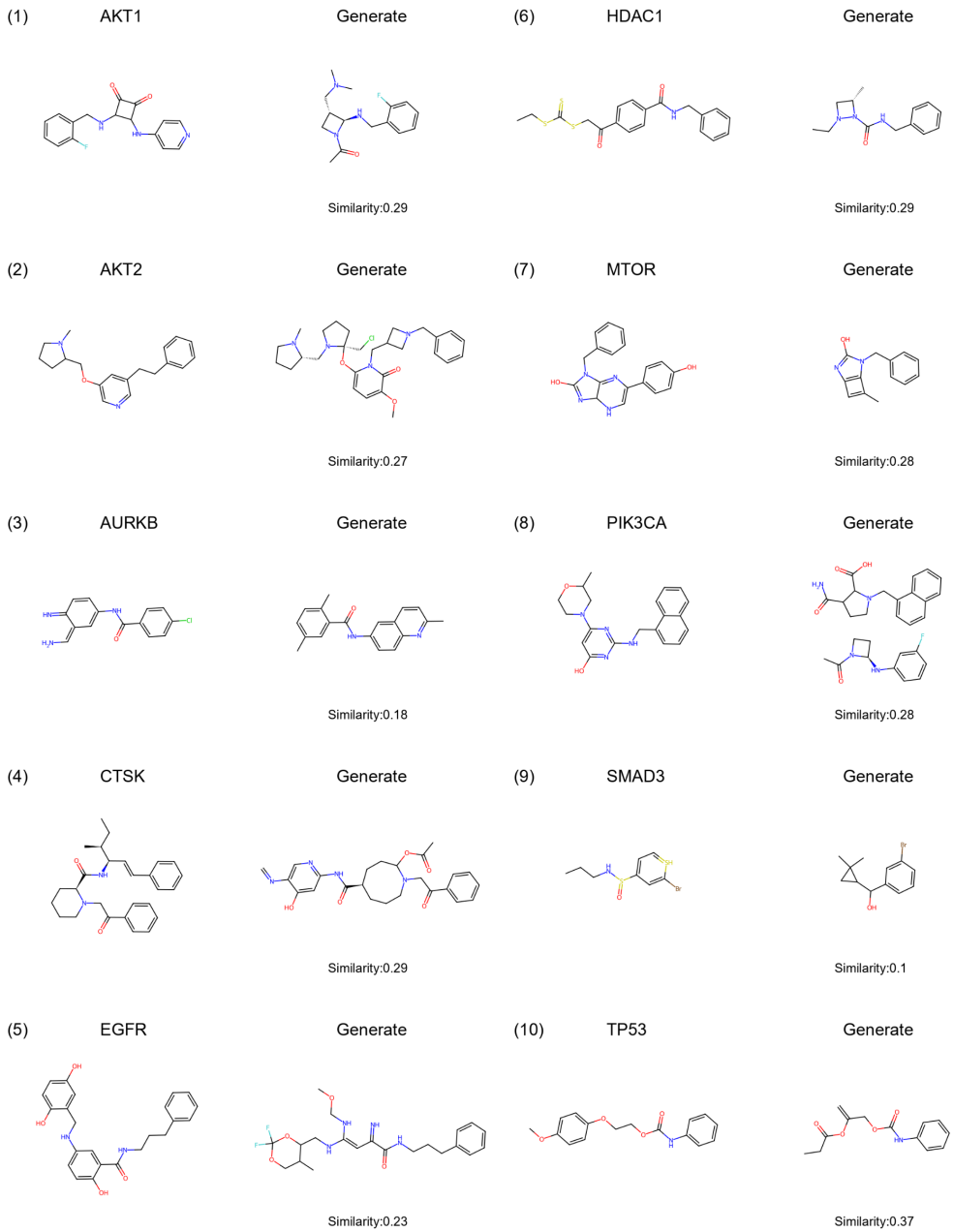


Fig. B1 This figure demonstrates examples of potential gene inhibitors, generated by WGAN [4], using knock-out gene expression signatures. The SMILES of molecules were extracted from Figure 2 in the WGAN article using OCR technology. Following this, we calculated the similarity results based on the Tanimoto similarity of Morgan fingerprints. Please refer to Table (labelled as Table A.5) for more detailed results.

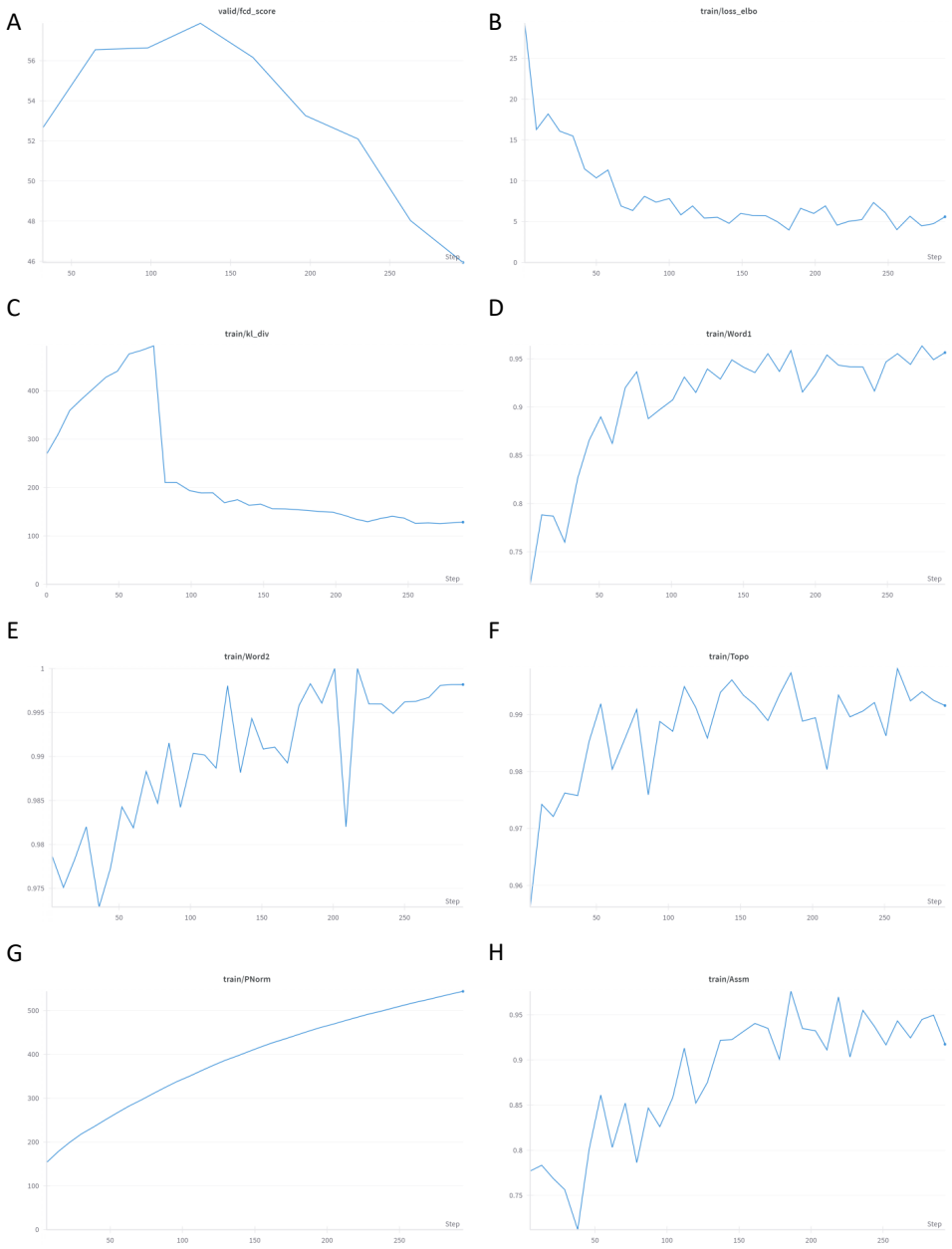


Fig. B2 This figure provides a visual representation of metric variation within the pre-training phase of the molecular hierarchically encoder-decoder. Section "A" highlights the variation of the FCD score [39], which is the primary metric used for model selection in this context. "B" and "C" symbolize the reconstruction and Kullback-Leibler Divergence (KL) loss respectively, with both demonstrating a decline as the training phase progresses. "D - H" denote the improving accuracy of the decoder at different layers throughout the training process.

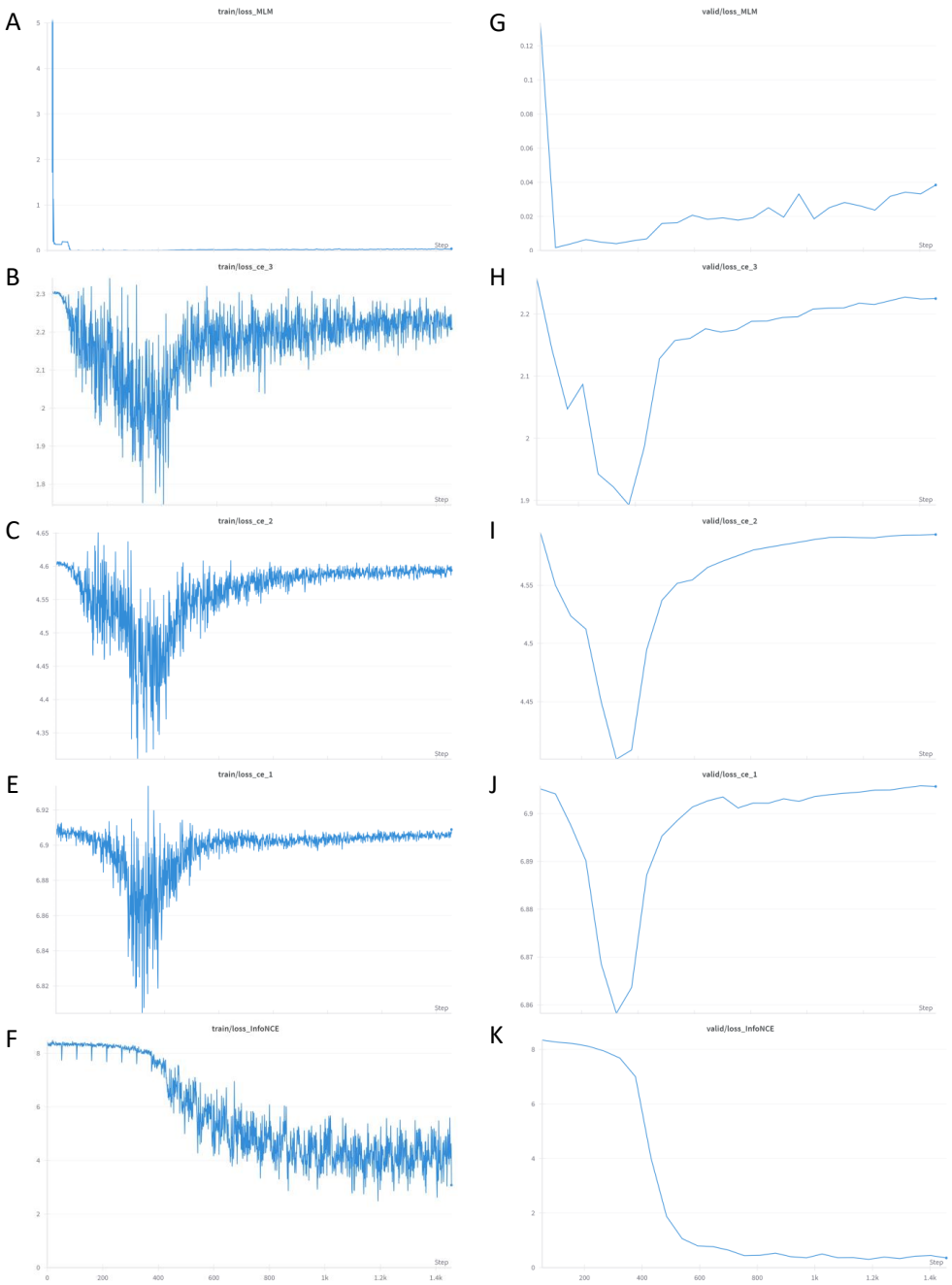


Fig. B3 This figure illustrates the variations in metrics during the alignment training process: Reports "A - F" detail the variations in MLM loss $\mathcal{L}_{\text{MLM}}$, auxiliary-guidance losses $\mathcal{L}_{\text{CE}_c}$, and InfoNCE loss $\mathcal{L}_{\text{InfoNCE}}$ on the training set, respectively. Reports "G - K" provide these details on the validation set.

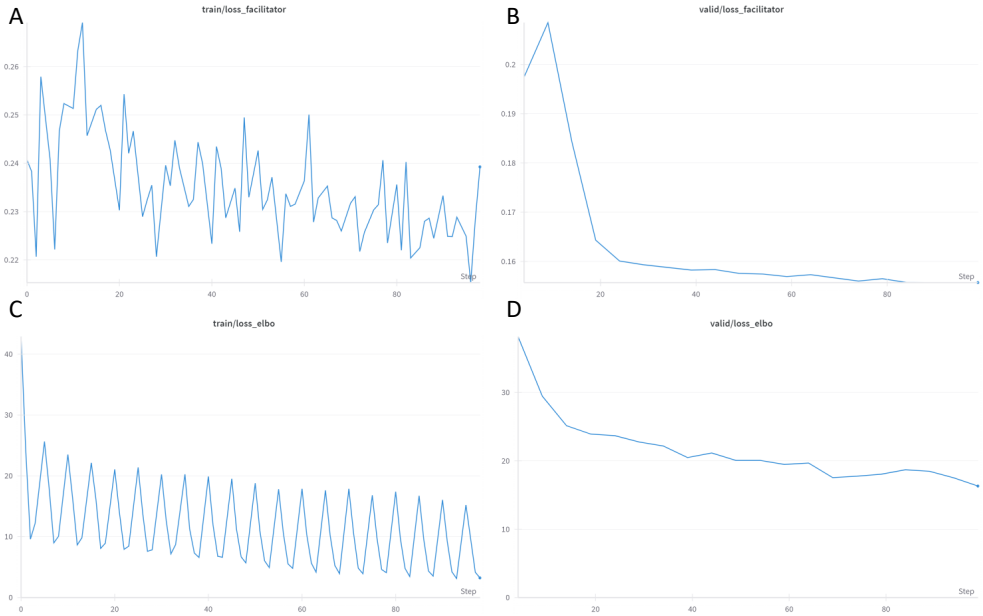
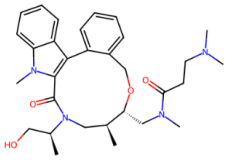


Fig. B4 This figure depicts variations in specific losses during two separate training processes. "A - B" illustrate the detailed changes in the auto-regression loss $\mathcal{L}_{\text{facilitator}}$ during the training process of the Facilitator (refer to Fig. 1D). Sections "C - D" show variations in the ELBO loss $\mathcal{L}_{\text{ELBO}}$ during the Molecular Decoder process (refer to Fig. 1E).

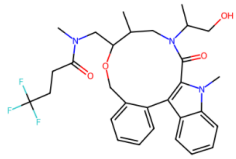
(1)

Reference



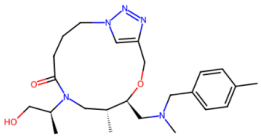
BRD-K99026889

Generate

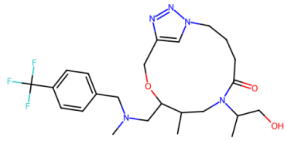


Similarity:0.81

(2)

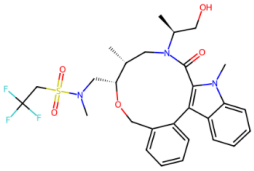


BRD-K54834493

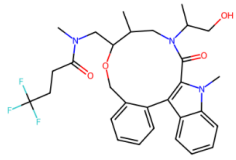


Similarity:0.81

(3)

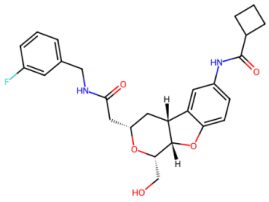


BRD-K50404714

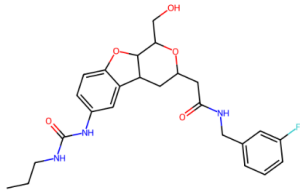


Similarity:0.77

(4)

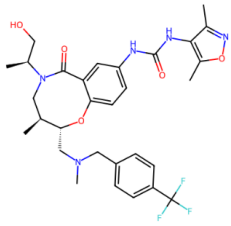


BRD-K36888933

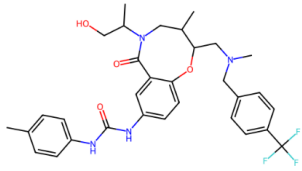


Similarity:0.74

(5)



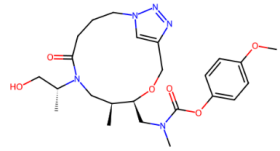
BRD-K76559855



Similarity:0.74

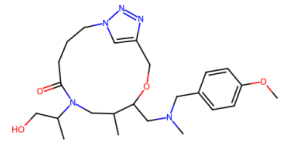
(6)

Reference



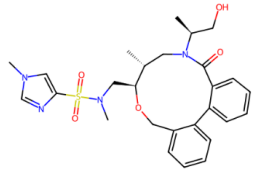
BRD-K37487131

Generate

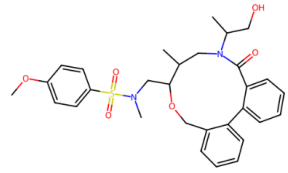


Similarity:0.71

(7)

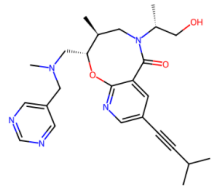


BRD-K25354424

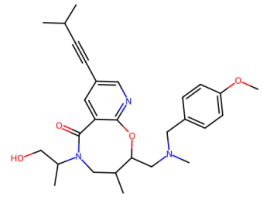


Similarity:0.71

(8)

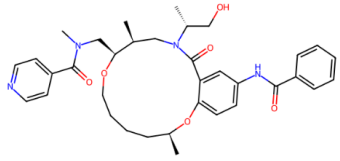


BRD-K22721982

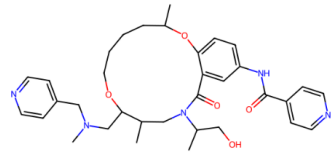


Similarity:0.7

(9)

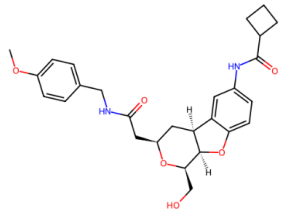


BRD-K73388776

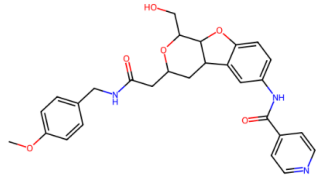


Similarity:0.7

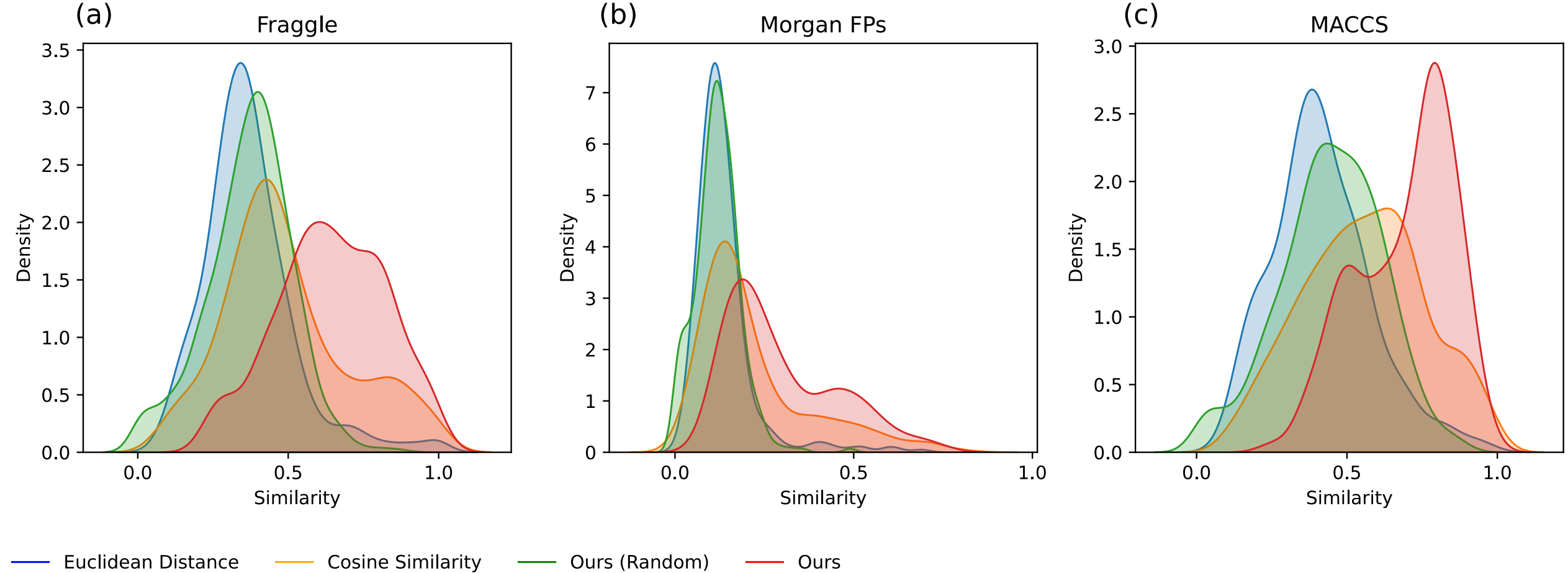
(10)



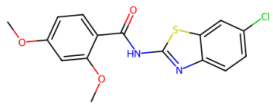
BRD-K83644384



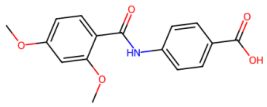
Similarity:0.7


(a)
Fraggle
(b)
Morgan FPs
(c)
MACCS
Density
Similarity
Euclidean Distance
Cosine Similarity
Ours (Random)
Ours

(1) AKT1 Generate

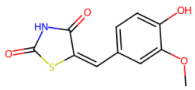


Kinome_2579

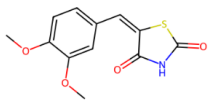


Similarity:0.45

(2) AKT2 Generate

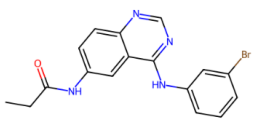


Kinome_3045

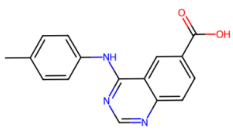


Similarity:0.65

(3) AURKB Generate

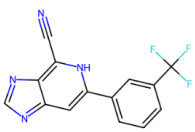


pd 174265

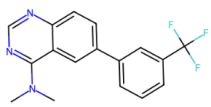


Similarity:0.43

(4) CTSK Generate

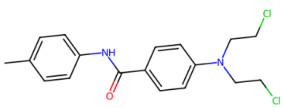


49862793

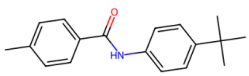


Similarity:0.38

(5) EGFR Generate

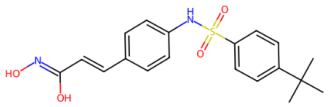


CHEMBL601040

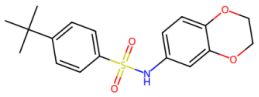


Similarity:0.49

(6) HDAC1 Generate

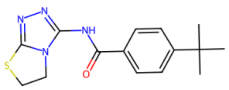


CHEMBL97272

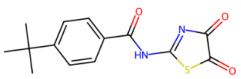


Similarity:0.44

(7) MTOR Generate

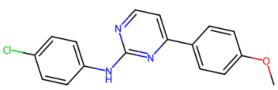


CHEMBL1594868

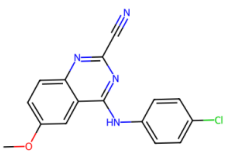


Similarity:0.45

(8) PIK3CA Generate

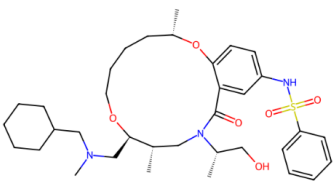


CHEMBL3134606

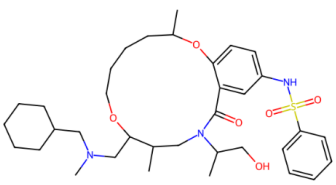


Similarity:0.39

(9) SMAD3 Generate

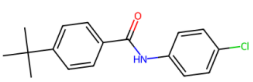


MLS003128856

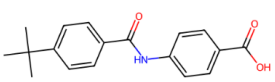


Similarity:1.0

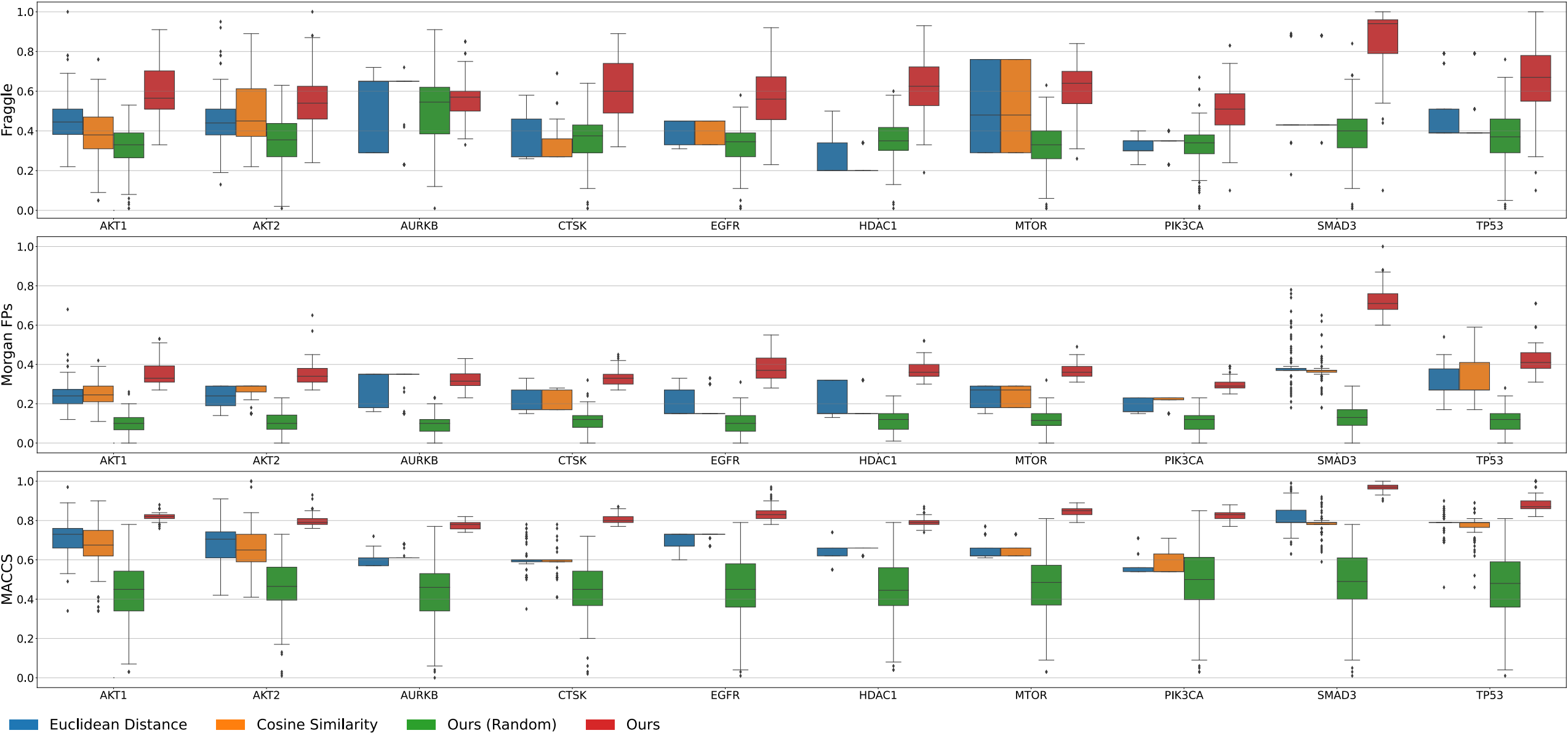
(10) TP53 Generate

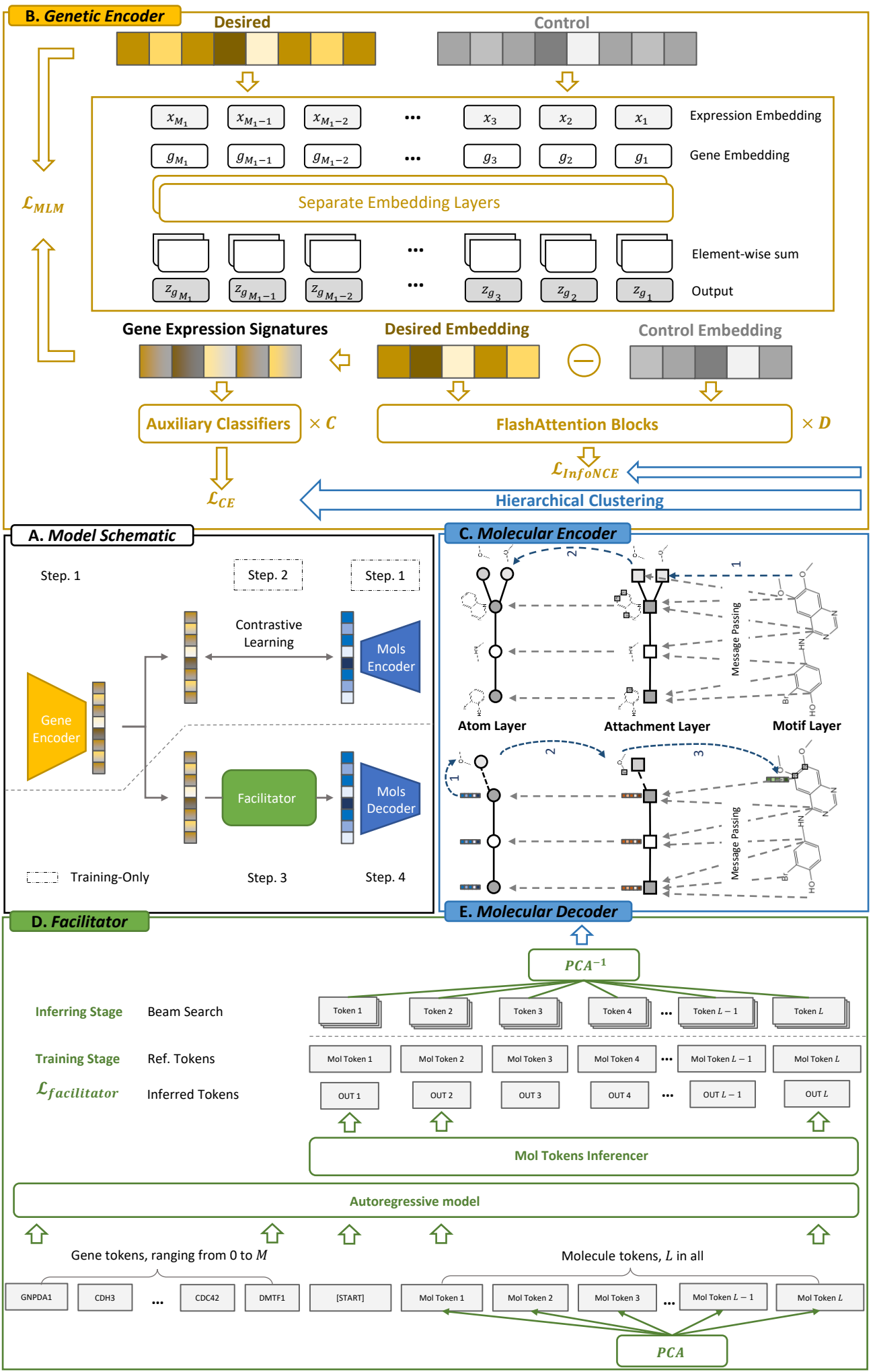


MLS000700014



Similarity:0.71


Fraggle
Morgan FPs
MACCS
1.0
0.8
0.6
0.4
0.2
0.0
AKT1
AKT2
AURKB
CTSK
EGFR
HDAC1
MTOR
PIK3CA
SMAD3
TP53
Euclidean Distance
Cosine Similarity
Ours (Random)
Ours


B. Genetic Encoder
Desired
Control
Expression Embedding
Gene Embedding
Separate Embedding Layers
Element-wise sum
Output
$\mathcal{L}_{MLM}$
Gene Expression Signatures
Desired Embedding
Control Embedding
Auxiliary Classifiers
× C
FlashAttention Blocks
× D
$\mathcal{L}_{InfoNCE}$
$\mathcal{L}_{CE}$
Hierarchical Clustering
A. Model Schematic
Step. 1
Step. 2
Step. 1
Contrastive Learning
Gene Encoder
Mols Encoder
Facilitator
Mols Decoder
Training-Only
Step. 3
Step. 4
C. Molecular Encoder
Atom Layer
Attachment Layer
Motif Layer
Message Passing
Message Passing
D. Facilitator
E. Molecular Decoder
PCA^{-1}
Inferring Stage
Beam Search
Training Stage
Ref. Tokens
$\mathcal{L}_{facilitator}$
Inferred Tokens
Token 1
Token 2
Token 3
Token 4
Token $L-1$
Token L
Mol Token 1
Mol Token 2
Mol Token 3
Mol Token 4
Mol Token $L-1$
Mol Token L
OUT 1
OUT 2
OUT 3
OUT 4
OUT $L-1$
OUT L
Mol Tokens Inferencer
Autoregressive model
Gene tokens, ranging from 0 to M
Molecule tokens, L in all
GNPDA1
CDH3
CDC42
DMTF1
[START]
Mol Token 1
Mol Token 2
Mol Token 3
Mol Token $L-1$
Mol Token L
PCA