ACS OMEGA

http://pubs.acs.org/journal/acsodf

Article

A Novel Scalarized Scaffold Hopping Algorithm with Graph-Based Variational Autoencoder for Discovery of JAK1 Inhibitors

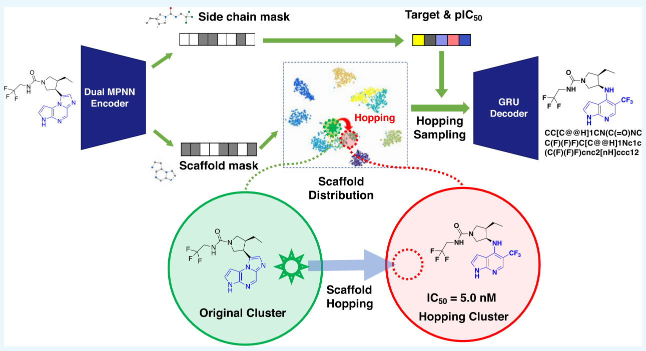
Yang Yu,[§] Tingyang Xu,[§] Jiawen Li, Yaping Qiu, Yu Rong, Zhen Gong, Xuemin Cheng, Liming Dong, Wei Liu, Jin Li, Dengfeng Dou,* and Junzhou Huang*

Cite This: *ACS Omega* 2021, 6, 22945−22954

Read Online

ACCESS | Metrics & More | Article Recommendations | SI Supporting Information

ABSTRACT: We have developed a graph-based Variational Autoencoder with Gaussian Mixture hidden space (GraphGMVAE), a deep learning approach for controllable magnitude of scaffold hopping in generative chemistry. It can effectively and accurately generate molecules from a given reference compound, with excellent scaffold novelty against known molecules in the literature or patents (97.9% are novel scaffolds). Moreover, a pipeline for prioritizing the generated compounds was also proposed to narrow down our validation focus. In this work, GraphGMVAE was validated by rapidly hopping the scaffold from FDA-approved upadacitinib, which is an inhibitor of human Janus kinase 1 (JAK1), to generate more potent molecules with novel chemical scaffolds. Seven compounds were synthesized and tested to be active in biochemical assays. The most potent molecule has 5.0 nM activity against JAK1 kinase, which shows that the GraphGMVAE model can design molecules like how a human expert does but with high efficiency and accuracy.

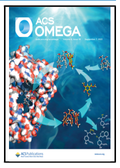


■ INTRODUCTION

Drug discovery is a time-consuming and costly process.[1,2] A major challenge in drug discovery is to identify novel candidate molecules toward specific targets quickly and accurately. Artificial intelligence has proven that it can be applied to solve this problem by fast identification of structurally novel ligands toward specific receptors.[3] It has been applied in several aspects in terms of conventional computer-aided drug design (CADD), such as virtual screening (ligand-based or structure-based), generative chemistry, drug repurposing, molecular docking, pharmacophore modeling, and so on. However, there are few deep learning algorithms for scaffold hopping methods as far as we know. The term scaffold hopping was coined by Gisbert Schneider and colleagues in 1999.[4] It refers to seeking for active molecules by replacing core structures of reference compounds. The main purposes for scaffold hopping in medicinal chemistry are bypassing the current intellectual property position, generating new lead series with better selectivity, and/or improving pharmacokinetic properties of known actives by modifying their chemical skeletons.

The traditional way to do scaffold hopping is by commercial software packages; however, they all suffer from a limited scaffold database, long computation time, or expensive licenses. Utilizing deep learning in drug discovery, especially in generative chemistry, has brought big progress and attention since Insilico Medicine reported their generative tensorial reinforcement learning (GENTRL) algorithm for identification of novel DDR1 inhibitors in weeks.[3] The advantages of generative chemistry are to explore a larger chemical space compared to virtual screening in current commercial compound libraries. Therefore, many endeavors are focused on developing molecular generation algorithms.[5−9] However, how to control the generation process at specific positions of lead compounds is very crucial for lead optimization in drug discovery. In this scenario, some key molecular skeletons should be kept unchanged from chemist perspectives. The valid molecular generation for scaffold hopping is just to keep side chains of the reference compound untouched and modify the scaffold. Although a few reports claimed that their deep learning algorithms can do scaffold hopping after that, limitations still exist. One recent work reported by Yang's group demonstrated a multimodal transformer algorithm ("DeepHop") by learning 50 K experimental molecular pairs across 40 kinases.[10] However, they did not define the scaffold of reference compounds that will be hopped. Therefore, the side chains that need to be retained will be changed instead of

Received: July 8, 2021
Accepted: August 9, 2021
Published: August 24, 2021



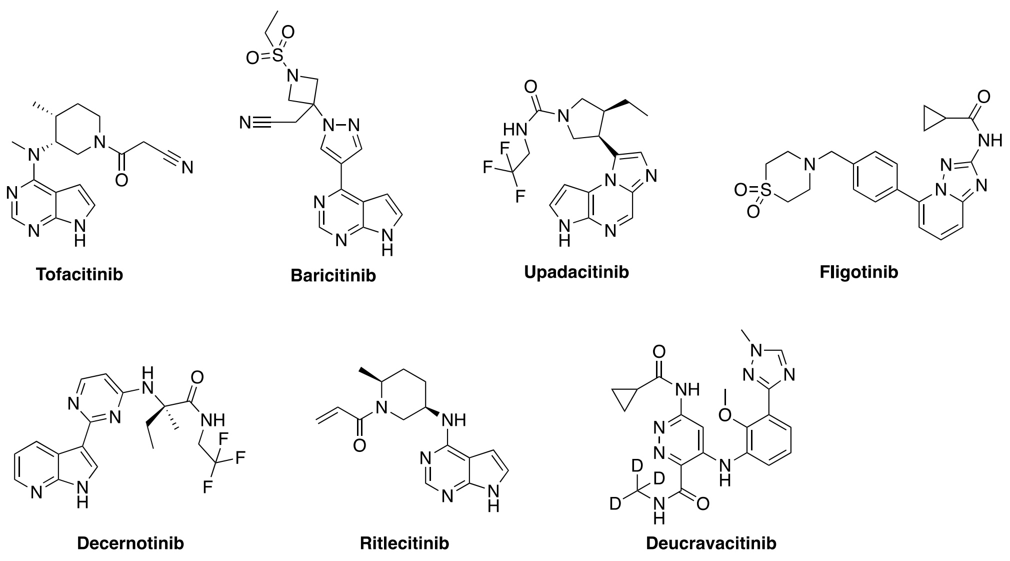


Figure 1. Seven FDA-approved JAKinibs drugs.

the desired scaffolds in real scaffold-hopping cases and even in their top-4 molecules of reported examples. A very promising work done by Deane's group developed a graph-based deep learning algorithm utilizing three-dimensional structural information ("DeLinker") for linker generation of two fragments.[11] Although they showed one scaffold-hopping case, the lack of experimental validation and the requisite for two side chains limit its further application in the scaffold-hopping scenario.

Janus kinases (JAKs) are a family of four non-receptor tyrosine kinases, which are JAK1, JAK2, JAK3, and TYK2. They and signal transducers and activators of transcription (STAT) proteins contribute to JAK−STAT pathways, which play a key role in the treatment of immunological and hematologic disorders. JAK inhibitors (JAKinibs) show to be efficacious in rheumatoid arthritis with sorts of adverse events, potentially due to the selectivity between different JAK isoforms.[12] JAKs were deemed as pivotal drug targets, and several JAKinibs have been approved by the FDA for the treatment of myeloproliferative neoplasms, graft versus host disease, rheumatoid arthritis, psoriatic arthritis, and inflammatory bowel disease.[13] Despite multiple clinical successes, some reported that side effects varied among JAKinibs. Therefore, the development of new JAKinibs still attracts great attention for reducing toxicity and improving efficacy. Currently, there are seven FDA-approved drugs against JAK and more than one hundred JAKinibs in clinical trials or discovery phases (Figure 1).

Herein, a novel deep generative algorithm (GraphGMVAE) for scaffold hopping is developed for the de novo design of new JAKinibs. To address the aforementioned limitations of current methods, several breakthroughs have been made:

a. We first incorporate ScaffoldGraph[14] and expert-made rules instead of the Bemis−Murcko scaffold algorithm to define and extract scaffolds.

b. From the computational viewpoint, scaffold hops can be classified into three categories with increasing dissimilarity between scaffolds: scaffold crawling (small-step hops), scaffold hopping (medium-step), and scaffold leaping (large-step).[15] However, no efforts have been made in controlling such a magnitude of scaffold hops from deep generative algorithms themselves. Therefore, we are the first to propose a controllable method that projects scaffolds into a latent space with Gaussian Mixture (GM) distribution so that the distance between two points in the latent space reflects the similarity between two scaffolds and each Gaussian kernel reflects one scaffold cluster. By selecting points from the Gaussian kernels of GM that are far away from the reference scaffold, the proposed method is able to choose scaffolds from the other scaffold clusters with different hopping controls.

c. A prioritizing pipeline was also proposed for molecular prioritization of the generated molecules by GraphGMVAE.

In this paper, we chose upadacitinib as the reference compound to showcase our algorithm that can rapidly and effectively design molecules that are distinct from known compounds in the literature and patents. 97.9% of the 30 K generated molecules were with novel scaffolds that are distinct from known JAKinibs. Seven of these molecules were designed, synthesized, and validated experimentally to show the potential applications for aiding chemists to quickly identify novel drug candidates.

■ RESULTS

Our proposed generative scaffold hopping model (GraphGMVAE) leverages the idea of style transferring. Style transferring is a kind of image generation technique that aims to generate a new image keeping the original content of a photo but changing its painting style to others, while a scaffold hopping task also requires keeping the original activity and the side-chain motif of the reference compound but swapping its scaffold to other structures.

Rule-Based Scaffold Extraction and Clustering. To build GraphGMVAE, the scaffolds of the molecules need to be extracted first. Bemis−Murcko scaffolds (BM scaffolds) are common and classic scaffold structures of molecules proposed by Bemis and Murcko. However, such scaffolds contain too many substructures for the model to learn scaffold patterns. Because the extraction method is just removal of all the substituents on the rings and retaining all the ring systems and

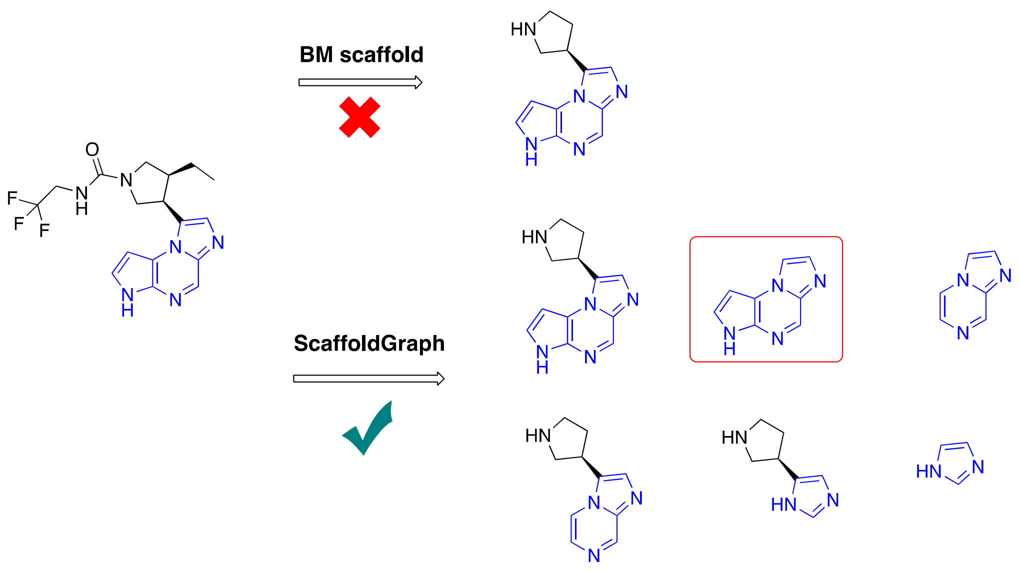


Figure 2. BM scaffold versus ScaffoldGraph: with BM scaffold, only side chains are removed to give a chemically meaningless scaffold, while with ScaffoldGraph, the desired scaffold can be obtained.

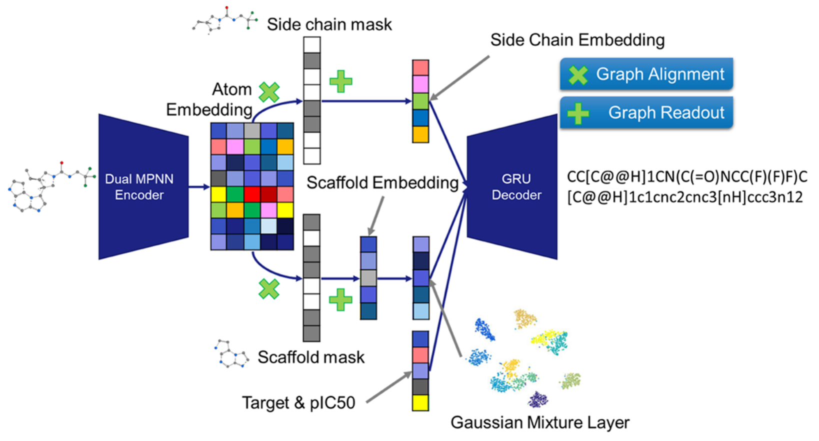


Figure 3. Overall diagram of GraphGMVAE: the Dual MPNN Encoder embeds the nodes information and topologies into node embeddings. Then, the side-chain embedding and scaffold embedding are read out from the node embeddings by masks. With the help of a Gaussian mixture layer, a resampled molecule embedding is fed to a GRU decoder to reconstruct back to corresponding SMILES. The details of Dual MPNN encoder and GRU decoder are in Dual MPNN and GRU Decoder for SMILES sections, respectively.

linkers between them, a new scaffold extraction named ScaffoldGraph[10] was employed, by which the secondary scaffolds were further extracted from BM scaffolds as shown in Figure 2. The BM scaffolds were first broken down into possible ring skeletons. Then, we filtered the scaffolds to get scaffold patterns from chemical perspectives by several structural rules. The rules can be extended and modified to meet different requirements. In general, three rules are made to define the scaffolds as follows.

(1) Ring counts $\geq$ 2: to remove some one-ring scaffolds, such as benzene, imidazole, cyclohexane, etc., due to their common and simple natures.
(2) Heavy atoms $\leq$ 20: the large scaffolds with too many rings will be filtered as well by the number of heavy atoms (>20 heavy atoms).
(3) Rotatable bonds $\leq$ 3: the scaffolds including too many rotatable bonds (>3) are also filtered to reduce the complexity of scaffolds.

The number of scaffolds is still huge after scaffold extraction. Moreover, many scaffolds are very similar to each other except for the presence of a few different atoms. Therefore, we apply spectral clustering on the extracted scaffolds using Tanimoto similarity as the affinity matrix (cutoff is 0.6). After spectral clustering, each scaffold is assigned a scaffold cluster ID.

Generative Scaffold Hopping Model (GraphGMVAE). The overall diagram of the proposed model is shown in Figure 3. We used dual message passing neural networks[16] (Dual-MPNN) to encode molecular structures and their properties. By connecting with a hidden space under Gaussian mixture distribution, we associated each scaffold cluster with a Gaussian distribution in the hidden space and pulled all scaffolds belonging to the same scaffold cluster to the same Gaussian centroid. Finally, a GRU-based SMILES decoder[17,18] was employed to decode the embeddings back to SMILES.

Since we hoped to hop the scaffolds but keep the side chain unchanged, we needed to set two different assumptions on both scaffold hidden space and side-chain hidden space. We assumed that the scaffold hidden space was under Gaussian mixture distribution but the side-chain hidden space was under Gaussian distribution. The assumption of a Gaussian mixture

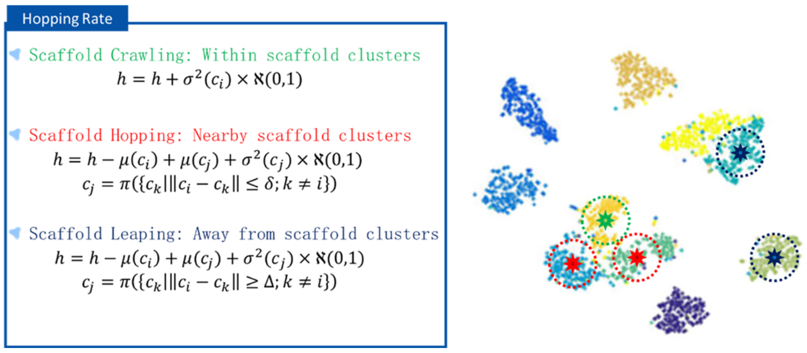

Hopping Rate
Scaffold Crawling: Within scaffold clusters
$h = h + \sigma^2(c_i) \times \aleph(0,1)$
Scaffold Hopping: Nearby scaffold clusters
$h = h - \mu(c_i) + \mu(c_j) + \sigma^2(c_j) \times \aleph(0,1)$
$c_j = \pi(\{c_k \| \|c_i - c_k\| \leq \delta; k \neq i\})$
Scaffold Leaping: Away from scaffold clusters
$h = h - \mu(c_i) + \mu(c_j) + \sigma^2(c_j) \times \aleph(0,1)$
$c_j = \pi(\{c_k \| \|c_i - c_k\| \geq \Delta; k \neq i\})$

Figure 4. Different hopping rates on Gaussian mixture distribution.

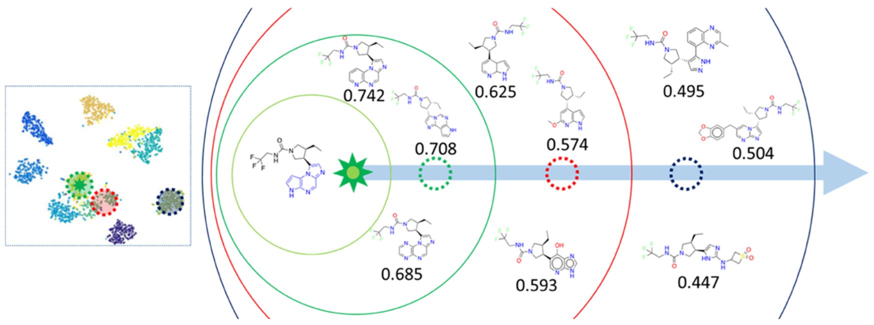


Figure 5. Generated exemplars in terms of the reference compound are sampled from the reference cluster (green circle), the nearby cluster (red circle), and the distant cluster (blue circle) via the Gaussian mixture distribution, respectively. The numbers under the molecules represent the Tanimoto similarities to the reference compound.

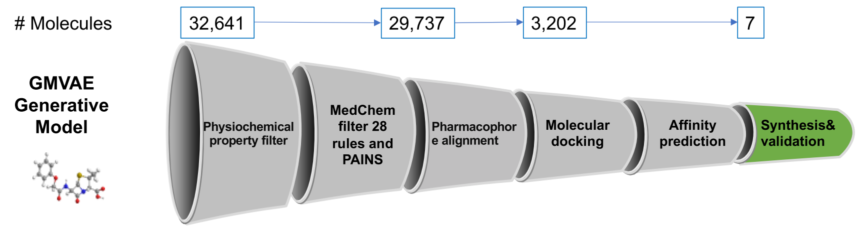


Figure 6. Prioritization pipeline for generative scaffold hopping.

on scaffold hidden space allowed us to hop the scaffold among Gaussian kernels (Figure 4).

Before training GraphGMVAE, we first made it learn a mapping of the chemical scaffold space, a set extracted from ZINC drug-like compounds (data set 1, 6.8 million) using the aforementioned scaffold extraction and clustering methods. This pre-training step enriched the diversity of scaffolds greatly compared to only extracting the skeletons from known kinase inhibitors. During training of the model, we then included a fine-tuning step to take into consideration molecular bioactivities, where we adopted a kinase data set of ~210 K small molecules with bioactivities ($pIC_{50} > 5$) from ChEMBL (data set 2). The details of the training model and molecular generation can be seen in the Methods section.

By sampling with different hopping rates, we generated three sets of new molecules from the reference cluster of the reference compound, the cluster near the reference cluster, and the cluster far away from the reference cluster. As shown in Figure 5, the different exemplars with increasing hopping rates have decreasing similarities to the reference compound.

Prioritization Pipeline for Generative Scaffold Hopping. To narrow down the generated molecules for evaluation, a prioritization pipeline was proposed (Figure 6). Tens of thousands of generated molecules were first filtered through drug-likeness and rule-based MedChem filters to remove potentially toxic molecules or molecules with reactive functional groups or poor drug-like properties. Herein, we only considered molecular weight (MW $\leq$ 550), octanol−water partition coefficient (CLogP $\leq$ 5), number of rotatable bonds (RotB < 10), and topological polar surface area (tPSA $\leq$ 120) instead of the common Lipinski rule of five (Ro5),[19] which medicinal chemists have adopted more. The MedChem filter included 480 PAINS filters (Pan Assay Interference Compounds)[20] and our 28 in-house rules. These rules were used for removing molecules containing potentially toxic substructures or reactive functional groups, such as furan, epoxide, Michael acceptors, etc.

The pharmacophore alignment was done by Forge software (https://www.cresset-group.com/software/forge/).[21] The pharmacophore template shown in Figure 7 was used to prioritize the molecules with the required pharmacophore for JAKinibs. A pharmacophore similarity score will be given by aligning the generated molecule with the template, and we set the threshold at 0.55 (the score is 0−1: 1 is the most similar; otherwise, the score is 0). Here, we only built the pharmacophore features around scaffolds of the reference

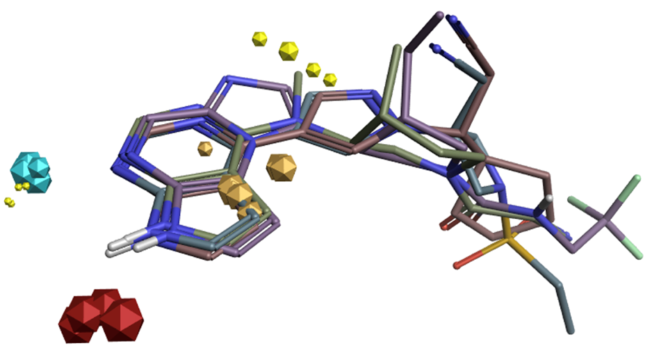


Figure 7. Pharmacophore template. The red field points (shown as diamonds) are positive or electrophilic; the cyan field points are negative or nucleophilic; the golden field points are hydrophobic; the light-yellow field points are van der Walls attractive. The size of the "diamonds" reflects the depths of each extremum energy well.

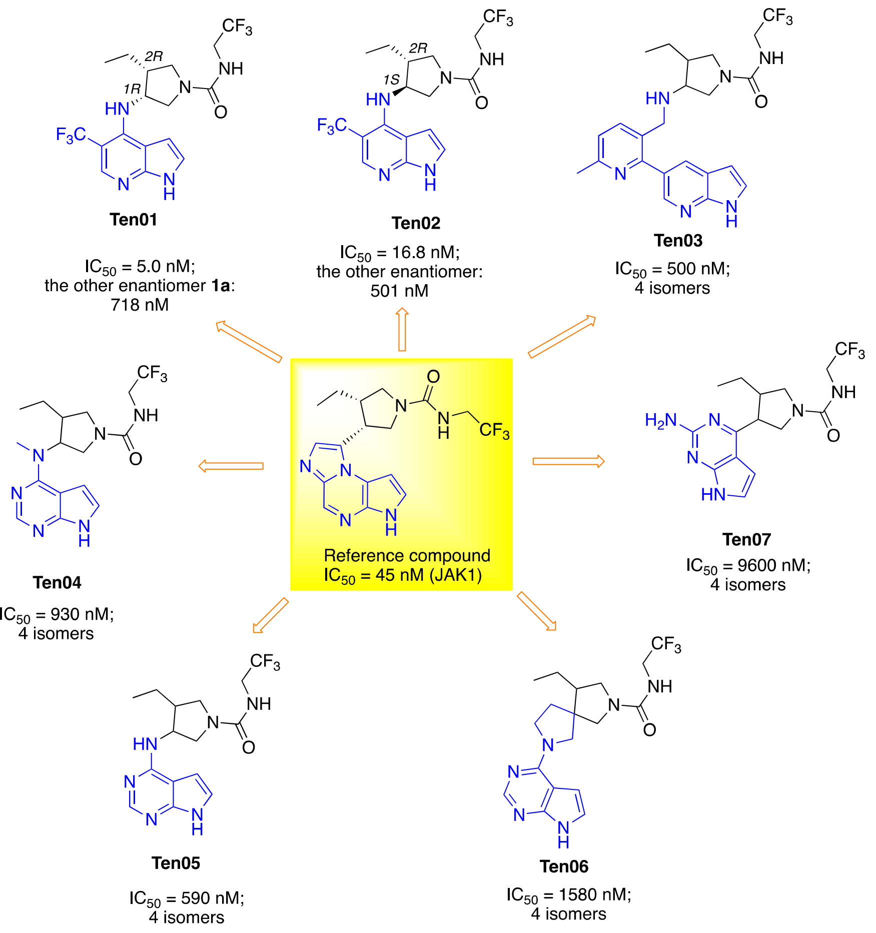
molecules to mitigate the effects of side chains on the score. By doing this, 3202 molecules were selected for molecular docking and affinity prediction.

The co-crystal structure (PDB code: 3EYG) of tofacitinib, a Janus kinase inhibitor, with the JAK1 kinase domain was optimized and minimized using Protein Preparation Wizard. The receptor grid was generated based on the centroid of tofacitinib's binding site with 20 Å buffer dimensions. Tofacitinib was redocked into the binding site with a docking score of −10.107 kcal/mol and RMSD value of 0.1099 Å. The docking score of upadacitinib with a rational binding pose of −10.268 kcal/mol indicates that this model is suitable for further research.

To further narrow down and prioritize the molecular docking results to a shortlist for visual checking of chemists, we utilized an in-house kinase 3D-CNN model[22−25] to predict the binding affinity between JAK1 and small molecules. The 3D-CNN model was trained using a dataset of inhibitors against 26 kinases deriving from ChEMBL (https://www.ebi.ac.uk/chembl/). The threshold rescore value for affinity prediction (CNNaffinity) was 7: that is to say molecules with a reasonable docking pose and CNNaffinity >7 will remain in the shortlist.

After the aforementioned pipeline, 25 molecules with high docking and CNN affinity scores were chosen for novelty check. Seven of them were selected for chemical synthesis based on synthetic feasibility followed by inhibition studies with JAK1. The structures of these seven compounds are shown below in Figure 8.

Compound Syntheses and JAK1 Inhibition. These seven compounds were synthesized (synthetic details in the Supporting Information), and the half-maximal inhibitory concentration (IC_{50}) values were measured in a typical ADP-Glo based kinase inhibition assay. Compound **Ten01** and its diastereomer **Ten02** showed potent JAK1 inhibition with IC_{50} values of 5.0 and 16.8 nM, respectively. It should be noted that **Ten01** and **Ten02** were separated from a mixture of four isomers by chiral resolution where the IC_{50} of the isomeric



Ten01
IC50 = 5.0 nM; the other enantiomer 1a: 718 nM
Ten02
IC50 = 16.8 nM; the other enantiomer: 501 nM
Ten03
IC50 = 500 nM; 4 isomers
Ten04
IC50 = 930 nM; 4 isomers
Reference compound IC50 = 45 nM (JAK1)
Ten07
IC50 = 9600 nM; 4 isomers
Ten05
IC50 = 590 nM; 4 isomers
Ten06
IC50 = 1580 nM; 4 isomers

Figure 8. Structures and JAK1 inhibition of the seven synthesized compounds.

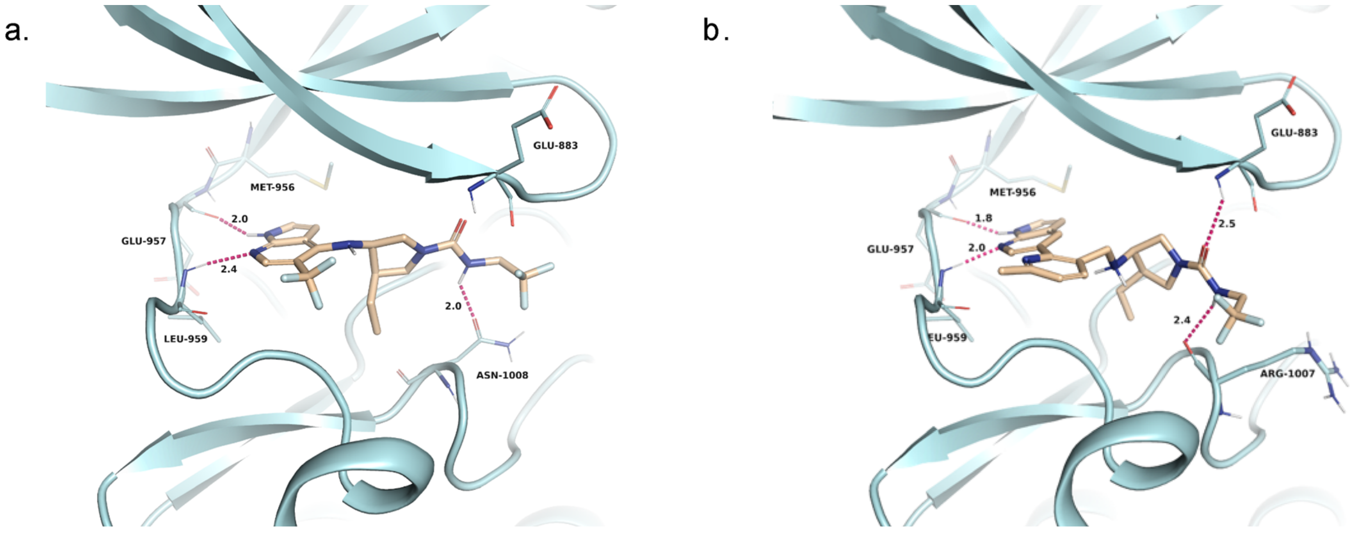
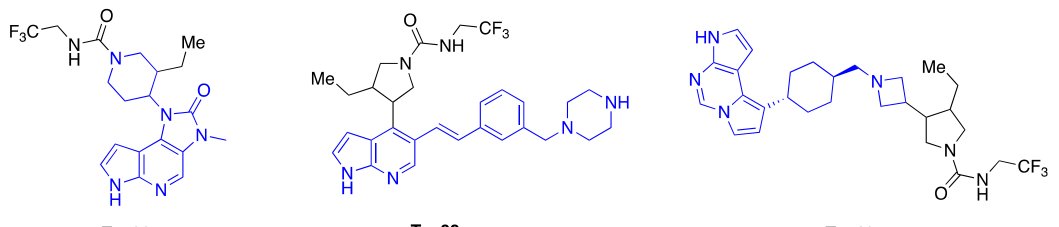


Figure 9. (a) Binding mode of compound (1*R*,2*R*)-**Ten01** (IC_{50} = 5.0 nM) in JAK1 (PDB code: 3EYG); (b) binding mode of compound (1*R*,2*R*)-**Ten03** in JAK1 (PDB code: 3EYG).



Ten08 Ten09 Ten10

Figure 10. Compound structures generated using BM scaffolds.

mixture is 64 nM. Compounds **Ten03**, **Ten04**, **Ten05**, **Ten06**, and **Ten07** exhibited good hit-level potencies (0.50, 0.93, 0.59, 1.58, and 9.6 μM, respectively). These **Ten03−07** isomeric mixtures are less potent compared to the **Ten01**/**Ten02** isomeric mixtures, and no chiral resolution was pursued although some of these chiral isomers may have IC_{50} values less than 100 nM. The binding modes of compounds **Ten01** and **Ten03** in JAK1 from docking simulations are shown in Figure 9.

■ DISCUSSION AND CONCLUSIONS

This study demonstrated a generative scaffold-hopping algorithm in a controllable way to design novel hit and lead compounds for the accelerating drug discovery process. By utilizing our generative model and pipeline, seven compounds are proven active against JAK1, two of which are even more potent than the reference compound. The two diastereomers were generated by our GraphGMVAE model, which means that the stereo information of small molecules was also considered. This shows that our deep learning algorithm is a rapid and successful way to identify new molecules with novel scaffolds and retains the biological activity of the reference compound toward the target.

To ease the chemical synthesis, this study aims to keep the side chain unchanged, although we can also change it by increasing the magnitude of scaffold hopping, which we called scaffold leaping. This requirement is just the challenge of previous scaffold hopping studies in that the algorithms cannot control which part of the reference compound to be hopped. To solve this, we are the first to incorporate the use of scaffold tree and expert rules to the generative model, which makes the generated molecules still drug-like with chemically defined scaffolds. Some bad cases that directly use the BM scaffold method instead are shown below in Figure 10. In compound **Ten08**, as the pyrrolidine ring is part of the BM scaffold, it was swapped for other ring systems as well as the desired tricyclic scaffold. For compounds **Ten09** and **Ten10**, the tricyclic scaffold was changed to undesired large BM scaffolds with many ring systems or with many rotatable bonds.

The second challenge for generative scaffold hopping is the scaffold diversity and novelty of generated molecules. Traditional scaffold-hopping software and previous algorithms are often limited by the libraries of scaffolds or training data and generating novel molecules but with high scaffold similarity to known inhibitors. By pre-training our GraphGMVAE model on ZINC 6.8 million drug-like molecules to learn more scaffold information and fine-tuning it with a kinase data set of about 210 K small molecules, we can successfully solve this challenge. It was shown that 97.9% of the scaffolds of generated molecules were novel compared to the training dataset by extracting BM scaffolds (only 471 molecules matched with known scaffolds).

The third challenge that needs to be solved for the deep-learning version of scaffold hopping is keeping the biological activities of generated molecules the same. The biological activities in terms of the targets were inputted as conditions of our generative model. It is evident from the results that the selected molecules are all active toward JAK1, although we used pharmacophore alignment and predicted binding affinity scores to prioritize the generated molecules.

The fourth challenge that has been solved is to control the magnitude of scaffold hops by selecting points from the Gaussian kernels of Gaussian Mixture (GM) distribution that are far away from the reference scaffold in which the distance between two points in the latent space reflects the similarity between two scaffolds.

Moreover, our study also shows that artificial intelligence can design novel molecules like how human experts can but in a more efficient and accurate manner. As we were doing chemical synthesis and biological validation of selected molecules, compound **Ten01** was published by a patent (WO2020182159A1) on September 17, 2020, two months after our algorithm generated the same molecule. We also double-checked this molecule, and its scaffold was not in our training data set. The scaffold was in accordance with our novelty check when we had prioritized this molecule to our final list in July 2020.

In conclusion, we demonstrated a generative scaffold hopping algorithm (GraphGMVAE) and a pipeline to prioritize molecules. As a proof of concept, upadacitinib was used as a reference compound to design inhibitors against the human Janus kinase 1 (JAK1) protein. Moreover, the designed molecules were experimentally synthesized and validated with their biological activities, proving the applicability of our model in drug discovery to find novel drug candidates like how human experts do but in a more efficient and accurate way. Furthermore, the incorporation of the ADMET prediction tool to the GraphGMVAE model to do lead optimization is under development.

■ METHODS

Dual-MPNN. The details of the structure of Dual-MPNN are shown in Figure 11. Dual-MPNN considers atom features

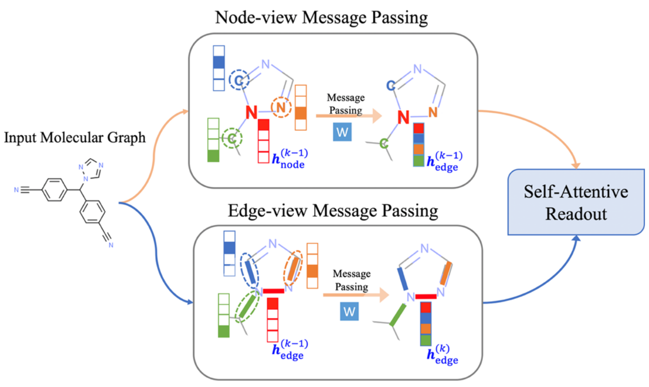


Figure 11. Structure of Dual-MPNN.

and bond features equally important for constituting a molecular representation vector based on its graph structure. As demonstrated in Figure 3, the model architecture contains two concurrent phases, Node-central Encoder and Edge-central Encoder, which are responsible for generating a node/edge embedding matrix from the graph topology and the node/edge features. We employ the message-passing neural network[26] as the backbone model to design the Node-central Encoder and Edge-central Encoder. Specifically, the Node-central Encoder is defined as:

$$\boldsymbol{m}_v^{l+1} = \sum_{u \in \mathcal{N}_v} f(\boldsymbol{h}_v^l, \boldsymbol{h}_u^l, \boldsymbol{e}_{vu}) \tag{1}$$

$$\boldsymbol{h}_v^{l+1} = \sigma(\boldsymbol{W}_{\text{node}} \boldsymbol{m}_v^{l+1} + \boldsymbol{h}_v^0) \tag{2}$$

where $f(\cdot)$ refers to an aggregation function and $\boldsymbol{h}_v^0$ is the input state calculated by a linear layer, $\boldsymbol{h}_v^0 = \sigma(\boldsymbol{W}_{nin}\boldsymbol{x}_v)$. The message-passing process in Node-central Encoder contains L steps. At $l + 1$ step, Node-central Encoder updates the state of node v by aggregating the previous state of its neighbor node $u \in \mathcal{N}_v$ as well as itself and takes the corresponding edge features $\boldsymbol{e}_{vu}$ as attached features to generate the new state of node v. Here, $\boldsymbol{W}_{\text{node}} \in \mathbb{R}^{d_{hid}\times(d_e+d_{hid})}$ is the weight matrix shared in all steps and $\sigma(\cdot)$ is the activation function. Without any specification, we use ReLU(x) = max $(0, x)$ as the activation function by default. Therefore, the final output of Node-central Encoder is represented as $\boldsymbol{H}_n = [\boldsymbol{h}_1^L, ..., \boldsymbol{h}_n^L] \in \mathbb{R}^{d_{hid}\times n}$.

Similarly, the Edge-central Encoder is constructed by the message-passing neural network:

$$\boldsymbol{m}_{vw}^{l+1} = \sum_{u \in \mathcal{N}_v \backslash w} f(\boldsymbol{h}_{vw}^l, \boldsymbol{h}_{uv}^l, \boldsymbol{x}_u) \tag{3}$$

$$\boldsymbol{h}_{vw}^{l+1} = \sigma(\boldsymbol{W}_{\text{edge}} \boldsymbol{m}_{vw}^{l+1} + \boldsymbol{h}_{vw}^0) \tag{4}$$

where $\boldsymbol{h}_{vw}^0 = \sigma(\boldsymbol{W}_{ein}\boldsymbol{e}_{vw})$ is the input state of edge (vw) and $\boldsymbol{W}_{ein} \in \mathbb{R}^{d_{hid}\times d_e}$ is the input weight matrix. The state vector is defined on edge $\boldsymbol{e}_{vw}$, and the neighbor edge set of $\boldsymbol{e}_{vw}$ is defined by all edges connected to the start node v except node w. The attached features are the node features $\boldsymbol{x}_u$. Edge-central Encoder also contains L steps, and one more round message passing on nodes is employed to transform edge-wise embedding to node-wise:

$$\boldsymbol{m}_v^{\text{out}} = \sum_{u \in \mathcal{N}_v} f(\boldsymbol{h}_{uv}^L, \boldsymbol{x}_u) \tag{5}$$

$$\boldsymbol{h}_v^{\text{out}} = \sigma(\boldsymbol{W}_{eout} \boldsymbol{m}_v^{\text{out}}) \tag{6}$$

where $\boldsymbol{W}_{eout} \in \mathbb{R}^{d_{\text{out}}\times(d_n+d_{hid})}$ is the weight matrix. Therefore, the final output of Edge-central Encoder is represented as $\boldsymbol{H}_e = [\boldsymbol{h}_1^{\text{out}}, ..., \boldsymbol{h}_n^{\text{out}}] \in \mathbb{R}^{d_{\text{out}}\times n}$. By concatenating both representations from Node-central Encoder and Edge-central Encoder, the total embedding for all nodes are denoted as $\boldsymbol{H}_{node} = \text{concat}(\boldsymbol{H}_n, \boldsymbol{H}_e)^{\text{T}}$.

Side-Chain Embedding and Scaffold Embedding. The hidden representation of a compound is disentangled into two portions: the side chain embedding and the scaffold embedding. To get these two embedding, we deploy a novel node masking and graph readout. First, the node embeddings from Dual-MPNN are separated into side-chain node embeddings and scaffold node embeddings by node masking, which records whether one atom node belongs to the scaffold or not. Namely, $\boldsymbol{s}_{sca} = [1, i \in \text{scaffold}; 0, i \notin \text{scaffold}]\ \forall\ i \in \{1,2, ..., n\}$, where n refers to the number of atoms in one compound. Second, we aggregate the node embeddings into subgraph embeddings via graph readout. We apply a graph attention readout to the side-chain atoms (nodes) and the scaffold atoms (node) to obtain side-chain embedding and scaffold embedding, respectively. Note that $\boldsymbol{H}_{node} \in \mathbb{R}^{n\times d}$ represents n node embeddings with $d = d_{out} + d_{hid}$ dimensions as a matrix. $\boldsymbol{H}_{node}[\boldsymbol{s}]$ indicates the selections of the rows in $\boldsymbol{H}_{node}$ by the vector $\boldsymbol{s} \in \mathbb{R}^n$. Then, the side-chain embedding and the scaffold embedding are derived as

$$z_{sca} = \text{softmax}(\boldsymbol{W}_1 \tanh(\boldsymbol{W}_2 \boldsymbol{H}_{node}^T[\boldsymbol{s}_{sca}]))\cdot \boldsymbol{H}_{node}[\boldsymbol{s}_{sca}] \quad (7)$$

$$z_{sc} = \text{softmax}(\boldsymbol{W}_1 \tanh(\boldsymbol{W}_2 \boldsymbol{H}_{node}^T[\bar{\boldsymbol{s}}_{sca}]))\cdot \boldsymbol{H}_{node}[\bar{\boldsymbol{s}}_{sca}] \quad (8)$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ refer to the trainable parameters of graph attention readout. Therefore, we define the space that all z_{sca} values belong to as the scaffold hidden space and the space that all z_{sc} values belong to as the side-chain hidden space.

Gaussian Mixture Layer. Assume that there are M scaffold clusters corresponding to M Gaussian distributions with $\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\sigma}_m)$ and $\boldsymbol{Z}_{sca}$ represents all scaffold embeddings of the molecules. Then, we define a novel loss, called Scaffold Cluster Loss ($\mathcal{L}_{scl}$), to compute the probability of each scaffold that belongs to all scaffold centroids with respect to the distance between scaffold embedding and Gaussian kernels as

$$\mathcal{L}_{scl} = \text{cross\_entropy}\left(\boldsymbol{d}_i^{adj} + \frac{1}{2}\left(\sum_m \boldsymbol{\sigma}_m\right), c_i\right) \quad (9)$$

where c_i refers to the scaffold cluster ID of the ith compound and $\boldsymbol{d}_i^{adj}$ refers to an adjusted distance vector containing the adjusted distances from the ith scaffold embedding of molecule against all Gaussian means $\boldsymbol{\mu}_m$, which is calculated as follows:

$$\boldsymbol{d}_i = \frac{1}{2}(\mathbf{1}_M^T z_{sca,i} - \boldsymbol{\mu}_M)\boldsymbol{\Sigma}_M^{-1}(\mathbf{1}_M^T z_{sca,i} - \boldsymbol{\mu}_M)^T \quad (10)$$

$$\boldsymbol{d}_i^{adj} = \boldsymbol{d}_i + \text{onehot}(c_i) \odot \epsilon \boldsymbol{d}_i, \quad (11)$$

where $\mathbf{1}_M$ refers to an all-ones vector with length M, onehot($\cdot$) refers to an operator of generating the one-hot representation of a scalar a scalar, $\odot$ refers to an operator of element-wise production, and ϵ refers to a small enhancement hyper-parameter.

Conditions for Biological Activities. We encode the biological activities in terms of a certain protein target as conditions of the VAE model. First, the protein target IDs, y_{ID}, are transferred into one-hot vectors with the lengths equal to the total number of protein targets as $y_p = \text{onehot}(y_{ID})$. Then, the values of pIC$_{50}$, v, are appended to that one-hot vector as the biological conditions of the inputs that are denoted as $\boldsymbol{y} = \text{Concat}(y_p, v)$.

GRU Decoder for SMILES. The target reconstruction is not a graph since there are no powerful enough graph decoders. Instead, we reconstruct molecules back to their canonical SMILES like a natural language processing (NLP) task since canonical SMILES can be considered as a kind of "molecular" language to translate each molecular graph into a unique symbol sequence. Here, we employ gated recurrent unit (GRU)[18] to process the SMILES. First, all canonical SMILES of molecules are converted into one-hot embeddings according to a symbol dictionary. Then, these one-hot embeddings are the targets of the decoder with the embeddings from hidden side-chain and scaffold spaces as the input of the decoder. Therefore, the GRU decoder for SMILES is derived as

$$\boldsymbol{f}_t = \sigma_g(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + b_f) \quad (12)$$

$$\boldsymbol{r}_t = \sigma_g(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1} + b_r) \quad (13)$$

$$\hat{\boldsymbol{h}}_t = \phi_h(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + b_h) \quad (14)$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{f}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{f}_t \odot \hat{\boldsymbol{h}}_t \quad (15)$$

where t refers to the index of the output sequence, $\boldsymbol{h}_0 = \text{Concat}(z_{sca}, z_{sc}, \boldsymbol{y})$, $\sigma(\cdot)$ refers to a sigmoid function, and $\phi_h(\cdot)$ refers to a hyperbolic tangent. Finally, the reconstruction loss computes the difference between one-hot embeddings of SMILES, as $\boldsymbol{X}$, and all the outputs of the GRU decoder as $\boldsymbol{H}$ as

$$\mathcal{L}_{recon} = \text{cross\_entropy}(\mathbf{X}, \boldsymbol{H}) \quad (16)$$

Therefore, the overall loss consists of scaffold cluster loss and reconstruction loss as

$$\mathcal{L} = \mathcal{L}_{scl} + \beta \mathcal{L}_{recon} \quad (17)$$

where β refers to a hyper-parameter of balancing two losses. The model takes preprocessed data, which consists of original SMILES, the corresponding scaffolds, and the scaffold cluster ID, as input.

Molecular Generation with Scaffold Hopping. When processing the molecular generation for scaffold hopping, a reference molecule is required as an input and the scaffold to be hopped in the reference molecule is required as well. First, the trained model converts the reference molecule and its scaffold to corresponding scaffold embedding and side-chain embedding in the hidden scaffold and side-chain space, respectively. Then, we re-sample a new scaffold embedding from the Gaussian mixture distribution of the hidden scaffold space. Different re-sampling strategies lead to different scaffold hopping results. We define three different strategies for scaffold crawling, scaffold hopping, and scaffold leaping. For scaffold crawling, we re-sample the embedding from the same Gaussian kernel (i.e., the same scaffold cluster) so that the scaffold will not be changed too much. The scaffold crawling is formulated as

$$z'_{sca} = z_{sca} + \boldsymbol{\sigma}_{c_i}^2 \times \mathcal{N}(0, 1) \quad (18)$$

where $\mathcal{N}(0, 1)$ refers to a normal distribution and $\boldsymbol{\sigma}_{c_i}^2$ refers to a learned variance of the c_i-th Gaussian kernel that the reference scaffold belongs to. For scaffold hopping, we re-sample the embedding from the Gaussian kernels that are close to the Gaussian kernel of the reference scaffold so that the scaffold will be changed to a nearby scaffold cluster. The scaffold hopping is formulated as

$$z'_{sca} = z_{sca} - \boldsymbol{\mu}_{c_i} + \boldsymbol{\mu}_{c_j} + \boldsymbol{\sigma}_{c_j}^2 \times \mathcal{N}(0, 1) \quad (19)$$

$$c_j = \pi(\{c_k | \|\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_k}\| \le \delta;\ k \ne i\}), \quad (20)$$

where the $\boldsymbol{\mu}$ terms refer to the centroids of corresponding Gaussian kernels, $\pi(\cdot)$ refers to a binomial sampling from a set, and δ refers to a distance threshold of Gaussian kernels to the reference Gaussian kernel. Similarly, for scaffold leaping, we re-sample the embedding from the Gaussian kernels that are far away from the reference Gaussian kernel to change the scaffold a lot. We formulate the scaffold leaping as

$$z'_{sca} = z_{sca} - \boldsymbol{\mu}_{c_i} + \boldsymbol{\mu}_{c_j} + \boldsymbol{\sigma}_{c_j}^2 \times \mathcal{N}(0, 1) \quad (21)$$

$$c_j = \pi(\{c_k | \|\boldsymbol{\mu}_{c_i} - \boldsymbol{\mu}_{c_k}\| \ge \Delta;\ k \ne i\}), \quad (22)$$

where Δ refers to another distance threshold of Gaussian kernels to the reference Gaussian kernel. Different kernel selections of scaffold crawling, scaffold hopping, and scaffold leaping will be illustrated below. When the scaffold embedding is re-sampled from Gaussian mixture distribution, the new scaffold embedding is fed to the GRU decoder with original side-chain embedding and biological conditions to generate novel molecules that suppose that only the scaffold of the reference molecule is changed.

■ DATA AND SOFTWARE AVAILABILITY

Pre-training Dataset (Dataset 1). For the pre-training step, we have preprocessed a dataset of chemical structures of standard drug-like compounds from the ZINC database (https://zinc15.docking.org/tranches/home/#) by scaffold extraction and clustering to get the information for scaffolds. A preprocessed data set consisted of original molecular SMILES (simplified molecular input line entry system) representations, the corresponding extracted scaffolds, and their scaffolds cluster IDs. The properties for the molecules, such as their binding protein targets and corresponding affinities (pIC_{50}, the negative log of the IC_{50} value), were set as dummy values (i.e., all zeros).

Fine-Tuning Dataset (Dataset 2). For the fine-tuning dataset, we adopted a kinase dataset of ~210 K small molecules with bioactivities ($pIC_{50} > 5$) from ChEMBL (https://www.ebi.ac.uk/chembl/).

GraphGMVAE Model. The data sets for model pre-training and fine-tuning are standard drug-like compounds from the ZINC15 database and ChEMBL. All experiments were implemented using the programming language Python version 3.6.0 with cuda version 10.0, using the packages pytorch version 1.1, horovod 0.21.3, and RDKit version 2019.03.3.0. The pre-training experiments were conducted using eight nodes with each containing 50 CPUs, 8 GPU cards (NVIDIA Tesla P40) (24 GB memory per card), and 256 GB system memory. The fine-tuning and generating experiments were conducted using a node with 10 CPUs, 1 GPU card, and 32 GB system memory.

Pharmacophore Alignment. The pharmacophore alignment was performed by Forge software (https://www.cresset-group.com/software/forge/). A pharmacophore template is built based on 3D structure information of four known JAK inhibitors from Drugbank (www.drugbank.ca), which are baricitinib (DB11817), upadacitinib (DB15091), ruxolitinib (DB08877), and tofacitinib (DB08895).

Molecular Docking. The molecular docking was performed using Schrödinger software. JAK1 kinase domain was optimized and minimized using Protein Preparation Wizard. The 3D structures of generated molecules were built with LigPrep.[27] Molecular docking was processed by the Glide[28] module with standard precision (SP) and extra precision (XP).

Kinase 3D-CNN Model. The dataset for model training and testing was collected from PDB (https://www.rcsb.org). It includes 1880 PDBs from 26 kinase targets. The details of 26 kinase targets can be found in the Supporting Information. All experiments were implemented using the programming language Python version 2.7 with cuda version 9.1, Ubuntu 16.04, RDKit version 2017.03.,1 and gnina.[23] The experiments were conducted using CPU (Intel Xeon Gold 5218 CPU, 2.30GHz, 64 cores), 2 GPU cards (NVIDIA Tesla V100, 16 GB memory per card), and 512 GB system memory.

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acsomega.1c03613.

Details of chemical synthesis, characterization data and biological activity data of selected compounds, examples of some deprioritized molecules, and summary of prioritization process (PDF)

32,641 generated molecules by the GraphGMVAE model (XLSX)

■ AUTHOR INFORMATION

Corresponding Authors

Dengfeng Dou − *HitGen Inc., Chengdu 610200 Sichuan, P. R. China*; Email: df.dou@hitgen.com

Junzhou Huang − *Tencent AI Lab, Tencent, Shenzhen 518057, P. R. China*; Email: joehhuang@tencent.com

Authors

Yang Yu − *Tencent AI Lab, Tencent, Shenzhen 518057, P. R. China*; orcid.org/0000-0003-2125-3229

Tingyang Xu − *Tencent AI Lab, Tencent, Shenzhen 518057, P. R. China*

Jiawen Li − *HitGen Inc., Chengdu 610200 Sichuan, P. R. China*

Yaping Qiu − *HitGen Inc., Chengdu 610200 Sichuan, P. R. China*

Yu Rong − *Tencent AI Lab, Tencent, Shenzhen 518057, P. R. China*

Zhen Gong − *HitGen Inc., Chengdu 610200 Sichuan, P. R. China*

Xuemin Cheng − *HitGen Inc., Chengdu 610200 Sichuan, P. R. China*

Liming Dong − *HitGen Inc., Chengdu 610200 Sichuan, P. R. China*

Wei Liu − *Tencent AI Lab, Tencent, Shenzhen 518057, P. R. China*

Jin Li − *HitGen Inc., Chengdu 610200 Sichuan, P. R. China*

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.1c03613

Author Contributions

§Y.Y.and T.X. contributed equally. The manuscript was written through contributions of all authors. All authors have given approval to the final version of the manuscript.

Notes

The authors declare no competing financial interest.

■ REFERENCES

(1) Paul, S. M.; Mytelka, D. S.; Dunwiddie, C. T.; Persinger, C. C.; Munos, B. H.; Lindorg, S. R.; Schacht, A. L. How to improve R&D productivity: the pharmaceutical industry's grand challenge. *Nat. Rev. Drug Discovery* **2010**, *9*, 203−214.

(2) Avorn, J. N. The $2.6 Billion Pill — Methodologic and Policy Considerations. *N. Engl. J. Med.* **2015**, *372*, 1877−1879.

(3) Zhavoronkov, A.; Ivanenkov, Y. A.; Aliper, A.; Veselov, M. S.; Aladinskiy, V. A.; Aladinskaya, A. V.; Terentiev, V. A.; Polykovskiy, D. A.; Kuznetsov, M. D.; Asadulaev, A.; Volkov, Y.; Zholus, A.; Shayakhmetov, R. R.; Zhebrak, A.; Minaeva, L. I.; Zagribelnyy, B. A.; Lee, L. H.; Soll, R.; Madge, D.; Xing, L.; Guo, T.; Aspuru-Guzik, A. Deep learning enables rapid identification of potent DDR1 kinase inhibitors. *Nat. Biotechnol.* **2019**, *37*, 1038−1040.

(4) Schneider, G.; Neidhart, W.; Giller, T.; Schmid, G. Scaffold-Hopping by Topological Pharmacophore Search: A Contribution to Virtual Screening. *Angew. Chem., Int. Ed.* **1999**, *38*, 2894−2896.

(5) Meyers, J.; Fabian, B.; Brown, N. *De novo* molecular design and generative models. *Drug Discovery Today* **2021**, S1359. and references cite in

(6) Gao, K.; Nguye, D. D.; Tu, M.; Wei, G.-W. Generative Network Complex for the Automated Generation of Drug-like Molecules. *J. Chem. Inf. Model.* **2020**, *60*, 5682−5698.
(7) Olivercrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular de-novo design through deep reinforcement learning. *Aust. J. Chem.* **2017**, *9*, 48.
(8) Kim, M.; Park, K.; Kim, W.; Jung, S.; Cho, A. E. Target-Specific Drug Design Method Combining Deep Learning and Water Pharmacophore. *J. Chem. Inf. Model.* **2021**, *61*, 36−45.
(9) Winter, R.; Montanari, F.; Steffen, A.; Briem, H.; Noó, F.; Clevert, D.-A. Efficient multi-objective molecular optimization in a continuous latent space. *Chem. Sci.* **2019**, *10*, 8016−8024.
(10) Zheng, S.; Lei, Z.; Ai, H.; Chen, H.; Deng, D.; Yang, Y.Deep Scaffold Hopping with Multi-modal Transformer Neural Networks. *Theor. Comput. Chem.* **2020**, ChemRxiv. Preprint. DOI: 10.26434/chemrxiv.13011767.v1
(11) Imrie, F.; Bradley, A. R.; Schaar, M.; Deane, C. M. Deep Generative Models for 3D Linker Design. *J. Chem. Inf. Model.* **2020**, *60*, 1983−1995.
(12) Traves, P. G. JAK Selectivity and the Implications for Clinical Inhibition of Pharmacodynamic Cytokine Signaling by Filgotinib, Upadacitinib, Tofacitinib, and Baricitinib. *Ann. Rheum. Dis.* **2021**, *74*, 865−875.
(13) Villarino, A. V.; Gadina, M.; O'Shea, J. J.; Kanno, Y. SnapShot: Jak-STAT Signaling II. *Cell* **2020**, *181*, 1696−1696.
(14) Scott, O. B.; Chan, A. W. E. ScaffoldGraph: an open-source library for the generation and analysis of molecular scaffold networks and scaffold trees. *Bioinformatics* **2020**, *36*, 3930−3931.
(15) Sun, H.; Tawa, G.; Wallqvist, A. Classification of Scaffold Hopping Approaches. *Drug Discovery Today* **2012**, *17*, 310−324.
(16) Ma, H.; Bian, Y.; Rong, Y.; Huang, W.; Xu, T.; Xie, W.; Ye, G.; Huang, J. *Multi-View Graph Neural Networks for Molecular Property Prediction*; arXiv, 2020, arXiv:2005.13607 [q-bio.QM].
(17) Chen, D.; Gao, K.; Nguyen, D. C.; Chen, X.; Jiang, Y.; Wei, G.-W.; Pan, F. Algebraic graph-assisted bidirectional transformers for molecular property prediction. *Nat. Commun.* **2021**, *12*, 3521.
(18) Chung, J.; Gulcehre, G.; Cho, K.; Bengio, Y. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*; arXiv, 2014, arXiv: 1412.3555 [cs.NE].
(19) Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. *Learning phrase representations using RNN encoder-decoder for statistical machine translation*; arXiv, 2014, arXiv:1406.1078v3 [cs.CL].
(20) Shultz, M. D. Two Decades under the Influence of the Rule of Five and the Changing Properties of Approved Oral Drugs. *J. Med. Chem.* **2019**, *62*, 1701−1714.
(21) Baell, J. B.; Holloway, G. A. New substructure filters for removal of pan assay interference compounds (PAINS) from screening libraries and for their exclusion in bioassays. *J. Med. Chem.* **2010**, *53*, 2719−2740.
(22) Cheeseright, T.; Mackey, M.; Rose, S.; Vinter, A. Molecular Field Extrema as Descriptors of Biological Activity: Definition and Validation. *J. Chem. Inf. Model.* **2006**, *46*, 665−676.
(23) Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. Protein−Ligand Scoring with Convolutional Neural Networks. *J. Chem. Inf. Model.* **2017**, *57*, 942−957.
(24) Pereira, J. C.; Caffarena, E. R.; Santos, C. N. Boosting Docking-Based Virtual Screening with Deep Learning. *J. Chem. Inf. Model.* **2016**, *56*, 2495−2506.
(25) Torng, W.; Altman, R. B. 3D deep convolutional neural networks for amino acid environment similarity analysis. *BMC Bioinf.* **2017**, *18*, 302.
(26) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural message passing for quantum chemistry. *Int. Conf. Mach. Learn. PMLR* **2017**, *70*, 1263−1272.
(27) *LigPrep*; Schrödinger, LLC: New York, NY, 2020.
(28) *Glide*; Schrödinger, LLC: New York, NY, 2020.