Applied Intelligence (2023) 53:25868–25880
https://doi.org/10.1007/s10489-023-04915-8

SELF-EdiT: Structure-constrained molecular optimisation using SELFIES editing transformer

Shengmin Piao[1] · **Jonghwan Choi**[1,2] · **Sangmin Seo**[1,2] · **Sanghyun Park**[1]

Accepted: 24 July 2023 / Published online: 12 August 2023

Abstract
Structure-constrained molecular optimisation aims to improve the target pharmacological properties of input molecules through small perturbations of the molecular structures. Previous studies have exploited various optimisation techniques to satisfy the requirements of structure-constrained molecular optimisation tasks. However, several studies have encountered difficulties in producing property-improved and synthetically feasible molecules. To achieve both property improvement and synthetic feasibility of molecules, we proposed a molecular structure editing model called SELF-EdiT that uses self-referencing embedded strings (SELFIES) and Levenshtein transformer models. The SELF-EdiT generates new molecules that resemble the seed molecule by iteratively applying fragment-based deletion-and-insertion operations to SELFIES. The SELF-EdiT exploits a grammar-based SELFIES tokenization method and the Levenshtein transformer model to efficiently learn deletion-and-insertion operations for editing SELFIES. Our results demonstrated that SELF-EdiT outperformed existing structure-constrained molecular optimisation models by a considerable margin of success and total scores on the two benchmark datasets. Furthermore, we confirmed that the proposed model could improve the pharmacological properties without large perturbations of the molecular structures through edit-path analysis. Moreover, our fragment-based approach significantly relieved the SELFIES collapse problem compared to the existing SELFIES-based model. SELF-EdiT is the first attempt to apply editing operations to the SELFIES to design an effective editing-based optimisation, which can be helpful for fellow researchers planning to utilise the SELFIES.

Keywords Deep learning · Drug design · Molecular optimisation · SELFIES · Levenshtein transformer

1 Introduction

Drug discovery is a challenging process to overcome the long struggle between humans and diseases. Discovering drug candidates requires a great investment of time and money [1, 2]. Traditional approaches are based on expert knowledge and experience searching large chemical libraries [3, 4]. However, these approaches are inefficient due to vast number of possible candidate molecules. The estimated number of potential drug-like compounds is between 10^{23} and 10^{60}, whereas the total number of synthesised compounds is approximately 10^8 [5]. To efficiently accelerate the conventional paradigm, recent studies have utilised deep generative models to address structure-constrained molecular optimisation tasks [6], aiming to improve the pharmacological properties of existing compounds by modifying their molecular structures. Moreover, recent advancements have led to the emergence of preliminary artificial general intelligence (AGI) capabilities in various content generation models. Meanwhile, tasks such as molecular optimisation, brain-inspired intelligence [7, 8], and neuromorphic computing [9, 10] contribute to the overall development of AGI and have practical applications in specific fields.

Shengmin Piao and Jonghwan Choi are contributed equally to this work.

✉ Sanghyun Park
sanghyun@yonsei.ac.kr

Shengmin Piao
sungmin630@yonsei.ac.kr

Jonghwan Choi
mathcombio@yonsei.ac.kr

Sangmin Seo
ssm6410@yonsei.ac.kr

[1] Department of Computer Science, Yonsei University, Yonsei-ro 50, Seoul 03722, Republic of Korea

[2] UBLBio Corporation, Yeongtong-ro 237, Suwon 16679, Gyeonggi-do, Republic of Korea

Several machine-readable representations of molecules have been developed to utilise various deep generative models. Widely used molecular representation methods include simplified molecular-input line-entry system (SMILES) [11], self-referencing embedded strings (SELFIES) [12], and molecular graph representations. The molecular graph representation is the most intuitive approach because it resembles Kekulé diagrams with atoms and bonds. In molecular graph representation, each molecule is depicted as an undirected graph, in which atoms are mapped to nodes and bonded to the edges. The molecular graph representations advantages include abundant structural information and high interpretability. However, graph-based deep generative models require significant storage space and memory for graph data processing, resulting in low efficiency of molecular generation [13]. In contrast, string-based representations, such as SMILES and SELFIES, enable efficient computations with relatively less storage. SMILES is an ASCII string that simplify atoms, bonds, and chemical structures using strict grammar. The SELFIES was designed to guarantee 100% chemically valid molecular generation by enforcing formal grammar rules (Fig. 1).

Using various optimisation techniques, many studies have proposed molecular optimisation models that efficiently generate new molecules with improved properties. These studies succeeded in generating novel molecules with improved properties. However, they did not consider the synthetic feasibility of the generated molecules, resulting in improved properties that were synthetically infeasible [6]. The synthesis of individual compounds typically involves experienced chemists who conduct several validations to assess their synthetic feasibility. However, with the growing number of compounds requiring estimation, new metrics have been designed to predict synthetic feasibility using trained models [14, 15]. To achieve property improvement and synthetic feasibility, recent studies have exploited scaffold-based generation and editing-based optimisation to slightly modify molecular structures while retaining property-related parts [16–20]. Although scaffold and editing-based approaches predominantly exploit molecular graph representations because of their high interpretability, a recent comparative study revealed that string-based representations, despite their complex grammar, do not exhibit any evident shortcomings when applied to molecule optimisation tasks [21]. In addition, the string-based representation exhibits a slightly higher generation efficiency.

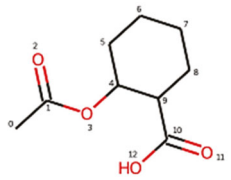
To design an effective SELFIES-based editing approach for structure-constrained molecule optimisation, the following criteria should be considered:

- *C1*: A tokenization method should be implemented to address the complex grammar of SELFIES.
- *C2*: The editing model should process molecular structural information at fine to coarse scales.
- *C3*: The outputs of the editing process must be chemically valid, property-improved, and structurally similar to the corresponding input molecules.

The existing string-based editing methods primarily tokenize molecules in atomic units [20]. Although this approach reduces the size of the token dictionary, it weakens the preservation of the structural features in the molecules. Furthermore, many SELFIES-based methods that utilise rule-based algorithms frequently suffer from a challenge known as SELFIES collapse during editing process [21]. SELFIES collapse refers to the phenomenon wherein different SELFIES strings containing grammatically incorrect substrings, collapse into a single truncated SELFIES string (Fig. 2). Owing to the collapse of SELFIES, SELFIES-based models have difficulty in generating diverse molecular structures.

Following the concept of fragment-based drug design [22], which involves utilising molecular fragments for stepwise optimisation, we propose a SELFIES Editing Transformer (SELF-EdiT) as a simple and efficient editing method based on SELFIES for structure-constrained molecule optimisation. To the best of our knowledge, SELF-EdiT is the first attempt to apply fragment-based editing operations to SELFIES. The main idea is to start with a seed molecule and generate candidates by deleting and inserting fragments of the SELFIES string. To ensure that these edits guaran-

(a)

Kekulé diagrams	
SMILES	CC(=O)OC1CCCCC1C(=O)O
SELFIES	[C][C][=Branch1][C][=O][O][C][C][C][C][C][C][Ring1][=Branch1][C][=Branch1][C][=O][O]

(b)

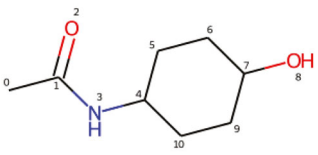
Kekulé diagrams	
SMILES	CC(=O)NC1CCC(O)CC1
SELFIES	[C][C][=Branch1][C][=O][N][C][C][C][C][Branch1][C][O][C][C][Ring1][#Branch1]

Fig. 1 Kekulé diagrams, SMILES, and SELFIES of (a) aspirin and (b) acetaminophen

(a)

SELFIES	[C][Branch2][Ring1][Branch2][Ring1][Ring1][C][N][C][=Branch1][C][=O][C][=C][C][Branch1][Ring1][O][C][=C][C][=C][Ring1][Branch2][Ring1][#Branch1]
Collapsed SMILES	C
Collapsed Molecule	CH_4

(b)

SELFIES	[C][Branch1][Branch2][Ring1][Ring2][C][N][C][=Branch1][C][=O][C][=C][C][=C][C][=C][Ring1][=Branch1][S][Ring1][=Branch2][Ring1][#Branch1]
Collapsed SMILES	C=O
Collapsed Molecule	

Fig. 2 Examples of SELFIES collapse. The alphabets "[Branch2]" and "[Branch1]" at the second position in (a) and (b), respectively, violate the SELFIES grammar rules

tee the structural information of the molecules, SELF-EdiT first segments SELFIES strings into fragment tokens based on the formal grammar of SELFIES (*C1*) and learns the embeddings of diverse fragment tokens by employing simple contrastive learning of sentence embeddings (SimCSE) [23] (*C2*). To edit a SELFIES string, SELF-EdiT uses a Levenshtein transformer (LevT) [24], which can iteratively perform deletion-and-insertion operations on SELFIES strings (*C3*).

2 Background

2.1 SELFIES

In SELFIES [12], atoms are represented by symbols enclosed in parentheses, such as [B], [C], [N], [O], [P], [S], [F], [Cl], [Br], and [I]. There are various types of bonds between atoms, including single, double, and triple bonds, which are represented as [-atom], [=atom], and [#atom], respectively. Based on these bonds, chemical structures such as branches and rings are produced and denoted by special symbols in SELFIES following specific grammar rules. The branches and rings are represented as [bond Branch N] and [bond Ring N], respectively, where N is the number of successors representing the length of the branch or ring.

As shown in (Fig. 1a), the branch of the 2nd oxygen bonded to the 1st carbon is written as "[C][=Branch1][C][=O]", indicating a double bond type of branch. Because the value of N is 1, the first character "[C]" on the right side of "[=Branch1]" is interpreted as an indicator, rather than a carbon. Using the SELFIES indexing table, we determined that this branch was a double-bond-type branch of length one.

Although the interpretation of the rings was similar to that of the branches, the search directions were different. For example, "[C][C][C][C][C][C][Ring1][=Branch1]" is a simple ring structure traversed from the 4th carbon and the 9th carbon (Fig. 1a). Because the value of N is 1, the first character "[=Branch1]" on the right side of "[Ring1]" is interpreted as an indicator to return the length of this ring structure.

Overall, the branches begin the search from the last successor and read sequentially based on their length, whereas the rings begin from the first predecessor and read backward.

2.2 Transformer

The Transformer [25] is an attention-based neural network architecture with an encoder-decoder structure. The encoder converts the input sequence as a sequence of hidden states, and the decoder decodes the output sequence from these hidden states. The decoding process is implemented in an autoregressive manner, where the next word in the sequence is predicted based on the previous words. The attention mechanism plays a crucial role in the transformer, allowing the model to learn complex syntax and capture long-range dependencies in the input sequence. This is essential for a sequence-to-sequence task requiring an understanding of the relationship between distant words.

The LevT [24] is a variant of a transformer that incorporats the ability to insert and delete tokens during the decoding process. A significant difference between LevT and the original transformer is the absence of a decoder in the former. In contrast, LevT incorporates a Levenshtein edit distance layer into the encoder, which calculates the Levenshtein distance [26] between the input and output sequences. This enables LevT to generate output sequences of varying lengths in a non-autoregressive manner, whereas the original transformer is constrained to output sequences of the same length as the input sequence. This capability is vital for tasks such as post-editing, in which output sequences of different lengths are required.

2.3 Related works

The existing methodologies for molecular optimisation can be grouped into four categories: 1) reinforcement learning, 2) Bayesian optimisation, 3) evolutionary algorithms, and 4) fragment-based optimisation. Reinforcement learning involves a generative model that randomly generates molecules, and an oracle that calculates the reward for the generated molecules based on their estimated molecular property scores. The generative model is then fine-tuned using a policy gradient algorithm with rewards to maximize the expected reward and generate desirable molecules [27, 28]. However, the use of a reinforcement learning scheme is

not straightforward because of the high variance in rewards [29].

Instead of fine-tuning generative models, such as reinforcement learning, molecular optimisation can be achieved by exploring latent chemical spaces. A popular exploration technique is the Bayesian optimisation method, which is normally coupled with a trained variational autoencoder (VAE) that learns the latent space corresponding to specific data in a probabilistic manner. More specifically, the process starts with known latent vectors of existing molecules with desired properties and structural similarities. Then, a surrogate model and acquisition function are updated to determine the direction that is most likely to optimise the seed molecules in the latent space [30, 31]. Although many studies have exploited Bayesian optimisation methods to explore the latent chemical space, designing an appropriate acquisition function can be challenging because of the high dimensionality and nonlinear features of the latent space [32].

Evolutionary optimisation approaches such as genetic algorithms and particle swarm optimisation determine the optimised molecular structures by fusing different molecules [20, 33]. The genetic algorithm is one of the most widely used evolutionary techniques and consists of two components: a set of mutation operations and a fitness function. This algorithm optimises molecules through mutations and/or crossover to perturb the mating pool containing a set of candidate molecules. At each iteration, new candidates are generated and evaluated based on their properties and structural similarities using a fitness function. As the algorithm progressed, unqualified candidates with low fitness scores were eliminated, allowing the most promising candidate molecules to survive in the mating pool and evolve towards an optimal molecule. However, it is worth noting that this approach can sometimes become trapped in regions of local optima [34].

While the previous three categories focus on optimisation algorithms, fragment-based optimisation focuses more on directly modifying the structure of the molecule. This approach can be traced back to the fragment-based drug design (FBDD) using traditional drug design methods [35]. Because fragments can be grown, merged, or linked to other fragments, FBDD optimises fragments by adding functional fragments or linking two independent fragments in an iterative process to improve their pharmacological properties [22, 36]. In line with this concept, recent studies have explored two main types of fragment-based optimisation: scaffold-based generation and editing-based optimisation. Scaffold-based generation represents a molecule as a tree of fragments that are then assembled in a fine-grained manner to optimise the molecules [16–19], whereas editing-based optimisation utilises addition and deletion operations to directly edit the internal fragments of the molecule [20].

3 Methods

In this section, we describe the overall process of SELFEdiT (Fig. 3), which consists of four steps: 1) SELFragment tokenization, 2) SELFragment embedding, 3) training procedure, and 4) inference procedure.

3.1 SELFragment tokenization

We first converted the original data from the SMILES representation (x_{smi}, y_{smi}) to the SELFIES representation (x_{sel}, y_{sel}), and then tokenized the SELFIES data into substructure-based fragments based on the SELFIES grammar. Because molecules can be considered combinations of branches and rings, the SELFIES strings can be tokenized into multiple fragments (SELFragments) that provide complete substructure information in accordance with the grammar.

Notably, because branches and rings are searched in different directions, a split operation must be performed twice during the tokenization process. More precisely, given a SELFIES string (Fig. 4a), the tokenizer first sequentially splits the string into branch-based fragments (Fig. 4b). However, relying solely on branch substructures for tokenization results in the ring substructure being either broken up or enclosed within a branch, leading to loss of molecular structural information. Hence, the tokenizer performs a backward search for the ring substructure in the obtained fragment sequence and rearranges the fragments via splitting or merging operations (Fig. 4c).

3.2 SELFragment embeddings

To efficiently deal with numerous SELFragments, we exploited SimCSE, an embedding model trained using contrastive learning. Contrastive learning aims to generate embedded representations that pull similar data closer to each other, while pushing dissimilar data far apart [37]. *Similarity* for contrastive learning should be well-defined depending on the task. SimCSE uses dropout masks as a data augmentation method to construct semantically similar positive pairs. Specifically, embeddings derived from identical inputs and dropout masks are regarded as positive instances, whereas embeddings derived from different inputs are treated as negative instances (Fig. 3). Specifically, for any SELFragment v, the objective function of SimCSE $L(v)$ is computed as

$$L(v) = -\log \frac{\exp(\mathrm{sim}(\xi^v, \xi^{v'})/\tau)}{\sum_{u \neq v} \exp(\mathrm{sim}(\xi^v, \xi^u)/\tau)}. \quad (1)$$

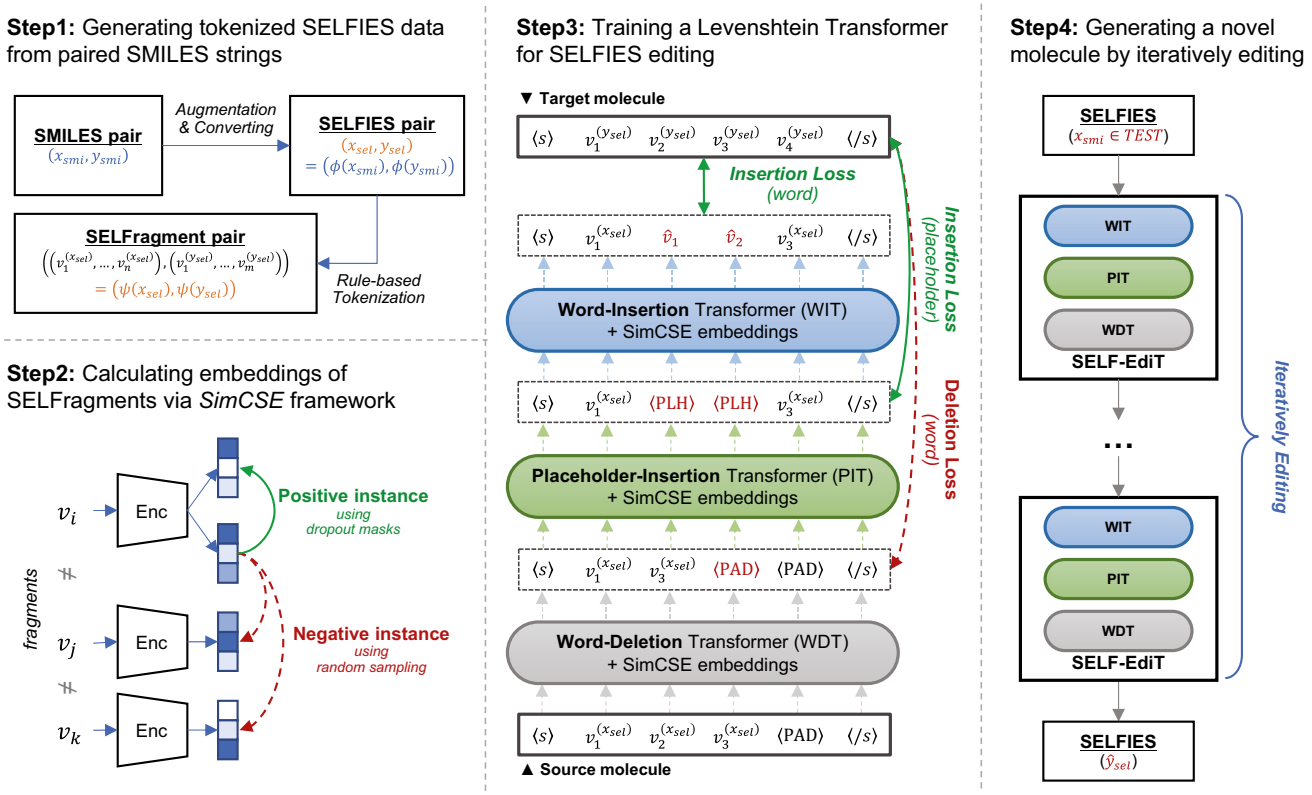


Fig. 3 Overall process of SELF-EdiT. The proposed method consists of four steps: rule-based SELFIES tokenization, contrastive learning for SELFIES fragment embedding, Levenshtein transformer for SELFIES editing operations, and molecular optimisation by iteratively editing SELFIES

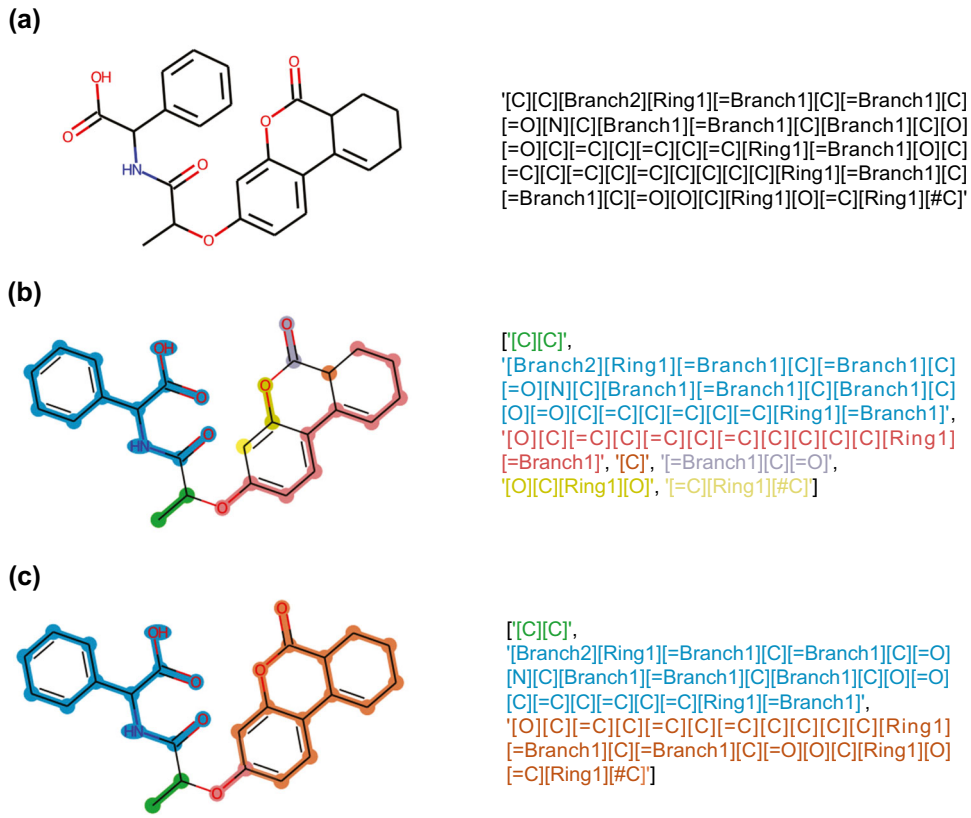


Fig. 4 Example of the rule-based SELFIES tokenization: (a) an initial SELFIES string; (b) tokenized SELFIES based on its branch symbols and grammar rules for branch; (c) rearranged SELFragments by considering ring symbols and grammar rules for ring; SELFragments and the corresponding structures are highlighted per colours

where v' is a SELFragment derived from v using a dropout mask, ξ^v is an embedding vector of v, τ is a temperature, and $\mathrm{sim}(a, b)$ is the cosine similarity $\frac{a^\top b}{\|a\|\|b\|}$.

3.3 SELF-EdiT

After tokenizing the input data pair and retrieving the corresponding embeddings, the model begins to edit the seed molecules. Toward this goal, we leveraged LevT, an edit-based neural machine translation model. LevT operates by starting with a source string and iteratively performing deletion and insertion operations on a sequence. During the training process, the deletion and insertion labels were obtained by calculating the Levenshtein distance between the source molecule x and the target molecule y. The Levenshtein distance is a string metric that efficiently measures the difference between two sequences using dynamic programming. Figure 3 shows the architecture of the SELF-EdiT, which consists of three transformers.

3.3.1 Word-Deletion Transformer (WDT)

The WDT first scans the input sequence $x = (< \mathrm{s} >, v_1, \cdots, v_{n-1}, < /\mathrm{s} >)$ and assigns a binary label l_i for each v_i. $l_i = 0$ indicates that the i-th fragment is retained and $l_i = 1$ indicates the deletion of the i-th fragment. The start token $< \mathrm{s} >$ and end token $< /\mathrm{s} >$ are excluded from the deletion process to ensure the integrity of the sequence boundaries. The WDT predictions were as follows:

$$\hat{l}_i^{del} = \mathrm{softmax}(h_i \cdot W_{\mathrm{WDT}}^\top) \quad \forall i = 1, \cdots, n-1, \tag{2}$$

where h_i is a hidden state of i-th SELFragment in x, $\hat{l}^{del}$ is a predicted deletion label, and $W_{\mathrm{WDT}} \in \mathbb{R}^{2 \times d_{model}}$.

3.3.2 Placeholder-Insertion Transformer (PIT)

After the WDT deletes the fragment in sequence x based on the deletion label, PIT predicts the number of tokens to be inserted into each adjacent fragment pair as follows:

$$\hat{l}_i^{plh} = \mathrm{softmax}(\mathrm{concat}(h_i, h_{i+1}) \cdot W_{\mathrm{PIT}}^\top) \quad \forall i = 0, \cdots, n-1, \tag{3}$$

where $v_0 =< \mathrm{s} >$, $\hat{l}^{plh}$ is a predicted integer for insertion of placeholder token $< \mathrm{PLH} >$, $W_{\mathrm{PIT}} \in \mathbb{R}^{(K_{max}+1) \times (2d_{model})}$, and K_{max} is the maximum number of placeholder token insertion count. The predicted number of $< \mathrm{PLH} >$ are inserted between fragment pairs to create a new masked sequence x'.

Algorithm 1 Training procedure.

$\mathcal{L} \leftarrow 0$
for $i \leftarrow 0$ to N **do**
  $l_i^{del} \leftarrow$ create_deletion_label (x_i, y_i)
  $\hat{l}_i^{del} \leftarrow$ WDT (x_i, y_i) ▷ (2)
  $\mathcal{L}_{del} \leftarrow$ NLL $(\hat{l}_i^{del}, l_i^{del})$ ▷ Negative Log-Likelihood loss
  $x_i^{del} \leftarrow$ delete_fragments (x_i, l_i^{del})
  $l_i^{plh} \leftarrow$ create_placeholders_label (x_i^{del}, y_i)
  $\hat{l}_i^{plh} \leftarrow$ PIT (x_i^{del}, y_i) ▷ (3)
  $\mathcal{L}_{plh} \leftarrow$ NLL $(\hat{l}_i^{plh}, l_i^{plh})$ ▷ Negative Log-Likelihood loss
  $x_i^{plh} \leftarrow$ insert_placeholders (x_i, l_i^{plh})
  $l_i^{ins} \leftarrow$ create_insertion_label (x_i^{plh}, y_i)
  $\hat{l}_i^{ins} \leftarrow$ WIT (x_i^{plh}, y_i) ▷ (4)
  $\mathcal{L}_{ins} \leftarrow$ NLL $(\hat{l}_i^{ins}, l_i^{ins})$ ▷ Negative Log-Likelihood loss
  $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_{del} + \mathcal{L}_{plh} + \mathcal{L}_{ins}$
end for
$\mathcal{L} \leftarrow \mathcal{L} \div N$
$Adam(\mathcal{L})$

3.3.3 Word-Insertion Transformer (WIT)

Given the masked sequence x' generated by PIT, WIT predicts the actual fragment for each inserted placeholder as follows:

$$\hat{l}_i^{ins} = \mathrm{softmax}(h_i \cdot W_{\mathrm{WIT}}^\top) \quad \forall i \in \{i \mid v_i =< \mathrm{PLH} >\}, \tag{4}$$

where $\hat{l}^{ins}$ is a predicted fragment that replaces the inserted $< \mathrm{PLH} >$ and $W_{\mathrm{WIT}} \in \mathbb{R}^{|V| \times d_{model}}$. Following LevT, instead of training modules with different weights, we implemented three modules that share the same transformer backbone to share useful features in different edit operations. In contrast to the original model, we tweaked the training process, as in Algorithm 1, to better align with our task.

3.4 Inference procedure

Once the training is completed, SELF-EdiT iteratively edits the source molecules in the format of SELFragments by alternating deletion and insertion operations. This procedure terminates when the modification count reaches a user-defined threshold and the optimal value is selected heuristically. Algorithm 2 outlines the inference process.

Algorithm 2 Inference procedure.

for $t \leftarrow 0$ to max_iter **do**
  $\hat{l}_i^{del} \leftarrow$ WDT (x_i, y_i) ▷ (2)
  $x_i^{del} \leftarrow$delete_fragments $(x_i, \hat{l}_i^{del})$
  $\hat{l}_i^{plh} \leftarrow$PIT (x_i^{del}, y_i) ▷ (3)
  $x_i^{plh} \leftarrow$insert_placeholders $(x_i, \hat{l}_i^{plh})$
  $\hat{l}_i^{ins} \leftarrow$WIT (x_i^{plh}, y_i) ▷ (4)
  $x_i^{ins} \leftarrow$insert_fragments $(x_i, \hat{l}_i^{ins})$
end for

4 Experiment

4.1 Datasets

The proposed method was trained and evaluated on two widely used benchmark datasets, dopamine receptor D2 (DRD2) and qualitative drug-likeness (QED), as described in [16]. Each datasets consists of training and testing sets, formed by pairwise data with specified property ranges and structural similarities $\text{sim}(x, y) \geq \delta$. Specifically, the DRD2 property score represents the probability that a compound is active against DRD2; these score values were evaluated using the trained model provided in [38]. In the DRD2 dataset, the source molecules had values < 0.05 and the paired target molecules had values > 0.5. The QED scores [39] measure how *druglike* a molecule is, and the open-source cheminformatics toolkit RDKit [40] was used to access the value. In the QED task, the goal was to optimise the source molecules in the range [0.7,0.8] to a higher range of [0.9,1.0]. To measure the structural similarity between the paired data, we utilised the Tanimoto similarity [41] over Morgan fingerprints and applied a similarity constraint $\delta = 0.4$ to all datasets.

We employed SMILES randomisation as the data augmentation method to enhance the efficiency of our model training. SMILES randomisation is a straightforward method that returns a diverse set of new SMILES strings by scanning a molecule starting from different atoms while retaining its structural integrity. We augmented as much data as possible to expand the training sets, whereas for the testing sets, each dataset was augmented 20 times to accommodate the quantitative analysis requirements.

4.2 Implementation details

Hardware: Nvidia GeForce RTX 3090 24GB GPU, Intel i9-9900K 3.60GHz CPU, and 32GB RAM. **Software:** Ubuntu 18.04.6 LTS, PyTorch 1.12.1, Python 3.7.12. For the hyperparameter settings of the SELFragment embeddings and SELF-EdiT, we adopted the default values used in the original SimCSE and LevT.

4.3 Baseline methods

We compared our SELF-EdiT model with the following baselines:

- **MMPA** [42]: A rule-based molecular transformation method that extracts several rules from a dataset. During the inference, the seed molecules are translated multiple times using different matching transformation rules.
- **Junction Tree VAE (JT-VAE)** [17]: A Bayesian optimisation-based model that represents the molecule graph as a junction tree that is cycle-free and easier to generate. The encoder maps both the molecular graph and junction tree into latent variables. The decoder first generated a junction tree as a blueprint, which was then reconstructed into a specific molecular graph.
- **GCPN** [27]: A reinforcement learning-based model that iteratively modifies a molecule by adding or deleting atoms and bonds. The proposed model also adopts adversarial learning to enhance the naturalness of optimised molecules.
- **VSeq2Seq** [30]: A Bayesian optimisation-based model that optimises SMILES-based sequences using VAE. Both the encoder and decoder use the GRU as the neural architecture and have been successfully applied to other molecule generation tasks.
- **UGMMT** [43]: A method that utilises dual learning to optimise molecules. To implement bidirectional conversion between the embedding spaces, each translation network was trained separately for one-way conversion.
- **VJTNN(+GAN)** [16]: An improved method based on the JT-VAE that treats molecular optimisation as a graph-to-graph translation task. The proposed method uses adversarial learning instead of Bayesian optimisation, while maintaining the junction tree encoder-decoder for learning.
- **HierG2G** [17]: A structural motif-based model that utilises a hierarchical graph encoder-decoder model to optimise molecules. The encoder generates a multi-resolution representation in a fine-to-coarse manner. Throughout the generation process, the autoregressive decoder progressively adds motifs in a coarse-to-fine manner.
- **T-S-Polish** [18]: A method proposes an optimisation paradigm called Graph Polish that aims to optimise molecules by maximizing the preserved portions of the source molecule through a Teacher and Student framework. The Teacher component identifies the optimisation center and provides information on the preservation, removal, and addition of other parts. The Student component learns this knowledge and applies it to optimise the molecules.
- **STONED** [20]: A rule-based method that edits SELFIES by replacing the source and target molecules. STONED has demonstrated its superiority in virtual screening for designing photovoltaic-like molecular structures and offers interpretability by drawing chemical paths from the source to the target molecules.

4.4 Molecular optimisation performance

To evaluate the overall optimisation performance of the proposed method, we compared SELF-EdiT with the baseline models using the following metrics:

- **Success** [16]: For each source molecule in a test dataset, the model generates K optimised molecules. We determined whether the optimisation was successful by checking if, among the K candidates, at least one molecule satisfied the similarity constraint and fell within the target range for the corresponding property. Finally, the success score was defined as the ratio of successful optimisation counts to the number of test molecules.
- **Property** [16]: The average property value of all the generated molecules.
- **Similarity** [16]: Average Tanimoto similarity between source molecules in the test dataset and corresponding generated target molecules.
- **Novelty** [16]: The proportion of molecules among the generated molecules that did not appear in the training sets. Novelty measures the potential of a model for the design of new molecules.
- **SF Score** (synthetic feasibility score): The proportion of molecules that simultaneously satisfy the similarity constraint, property improvement, and synthetic feasibility among the overall generated molecules. The synthetic feasibility was measured using GASA [15], a prediction framework that evaluates the synthetic feasibility of small molecules by classifying them as either 0 (easy to synthesize) or 1 (hard to synthesize).
- **Total Score**: The weighted sum of the SF score, property, similarity, and novelty, which comprehensively reflects the overall performance of the model. To properly consider the property improvement, structural similarity, and synthetic feasibility captured by the SF score, we assigned a weight of 1.5 to the SF score a weight of 1 to the other metrics.

We first calculated the success scores of SELF-EdiT and the baseline models because the success score evaluates the model performance in generating molecules with both property improvement and high structural similarity. As shown in Fig. 5, SELF-EdiT exhibited the highest success scores of 0.596 and 0.822 for QED and DRD2, respectively, compared with the baseline models. This demonstrates that SELF-EdiT is an effective tool for structure-constrained molecular optimisation.

To evaluate the overall generative performance of SELF-EdiT, we compared the total scores of the SELF-EdiT and baseline models (Fig. 6). SELF-EdiT outperformed the baseline models in terms of the total scores, achieving scores of 2.791 and 2.520 for QED and DRD2, respectively. The total scores of SELF-EdiT were ranged from 0.035 to 0.884, which were higher than the baselines, demonstrating the effectiveness of SELF-EdiT for structure-constrained molecular optimisation. To better understand the performance differences between the SELF-EdiT and baseline models, all metric scores are provided in Supplementary Tables S1 and S2. Although SELF-EdiT was not the best for each

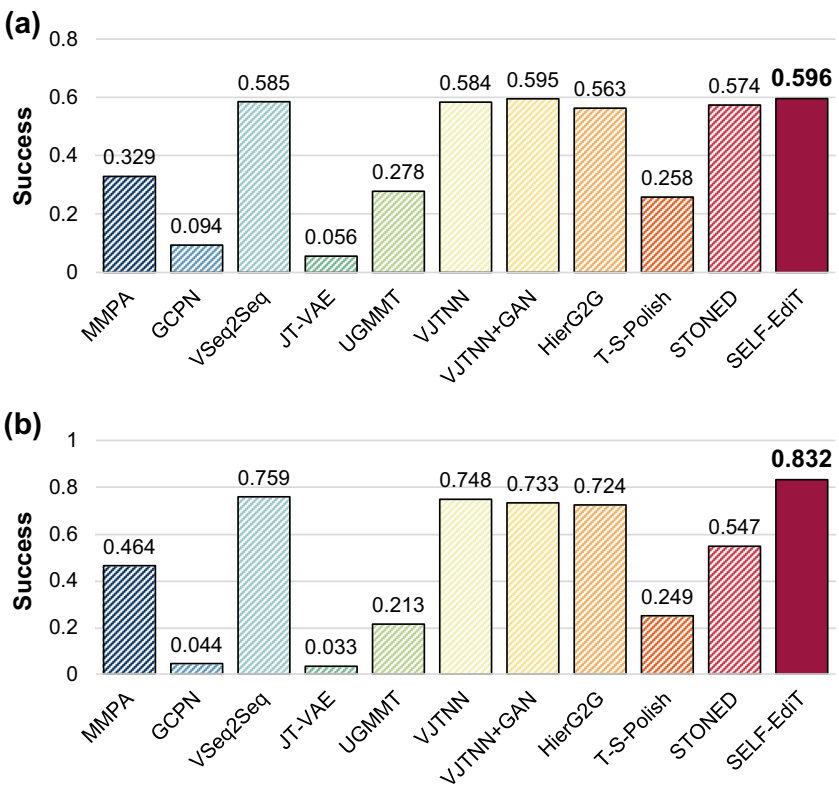


Fig. 5 The success scores of the molecular optimisation performance on (a) QED and (b) DRD2 datasets. The x-axis and y-axis indicate the success scores and the baseline models, respectively

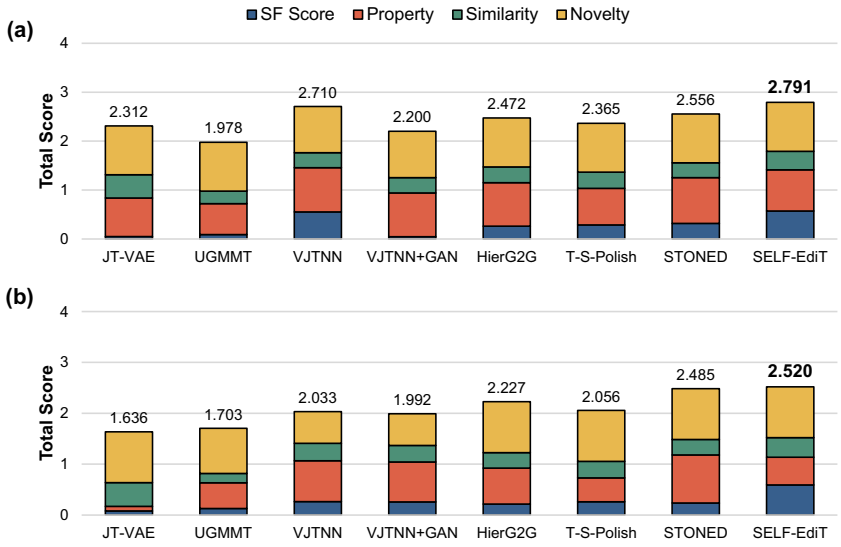


Fig. 6 The total scores of the molecular optimisation performance on (a) QED and (b) DRD2 datasets. The x-axis and y-axis indicate the total scores and the baseline models, respectively

property or similarity metric, it outperformed the baseline by achieving the highest SF scores. This indicates that SELF-EdiT can fulfill the requirements of property improvement and structural similarity while ensuring synthetic feasibility during molecular editing operations. These experimental results demonstrate that SELF-EdiT has the most balanced performance in structure-constrained molecular optimisation tasks.

4.5 Hyperparameter analysis

The quality of the output molecules generated by SELF-EdiT varies depending on the number of edit iterations. To investigate the optimal number of edit iterations, we compared the success rates evaluated for different edit count values. We conducted experiments by generating molecules with edit counts ranging from one to five. As shown in Fig. 7, the best success score for the DRD2 task was confirmed when the edit was performed twice (Fig. 7b), whereas in the case of the QED task, there was no significant difference in the success scores across the different edit count values (Fig. 7a). Based on these results, molecular editing with a high number of iterations may lead to low efficiency. For structure-constrained molecular optimisation, we identified an optimal edit count value of two.

4.6 SELFIES collapse evaluation

We confirmed that our edit-based SELFIES optimisation approach is more effective in mitigating SELFIES collapse than existing SELFIES-based approaches. Based on the characteristics of the SELFIES, any syntax conflicts are skipped during the conversion process to SMILES, leading to the collapse of the SELFIES. Therefore, we measured the SELFIES collapse rate by reconstructing the SELFIES strings using an official SELFIES-SMILES converter [12]. We decoded each given SELFIES string into a corresponding SMILES string and then encoded the SMILES string back into a SELFIES string. If the given SELFIES string was grammatically correct, the original and reconstructed SELFIES strings would be equal, and we can conclude that there was no collapse. Based on the above method, we computed and compared the Levenshtein distance-based collapse rates of SELF-EdiT and STONED, a state-of-the-art SELFIES optimisation method, on two property test sets (Figs. 8–9). Compared to the collapse rates of STONED on the QED and

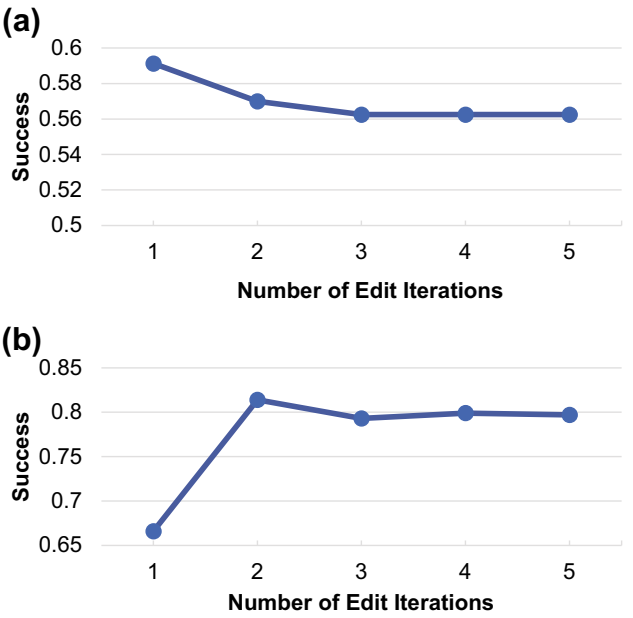


Fig. 7 Success scores for each number of edit iterations. (a) QED and (b) DRD2. The x-axis indicates the number of edit iterations and y-axis indicates the success scores

(a)

Source	[C][=C][C][Branch1][=Branch2][O][C][C][NH1+1] [Branch1][C][C][C][=C][C][=C][Ring1][N][C][N][C] [=Branch1][C][=O][C][C][O][C][=C][C][=C][Branch1][C][F] [C][=C][Ring1][#Branch1][C][Ring1][O]
Target	[C][=C][C][Branch1][#Branch2][O][C][Branch1][C][F] [Branch1][C][F][F][=C][C][=C][Ring1][O][C][N][C] [=Branch1][C][=O][C][C][O][C][=C][C][=C] [C][=C][Ring1][=Branch1][C][Ring1][#Branch2]
Levenshtein distance	**11**

(b)

Source	[O][=C][Branch1][O][N][C][Branch1][C][C] [Branch1][C][C][C][#C][C][=C][C][=C][Branch1] [Ring1][O][C][C][Branch1][C][O][=C][Ring1][=Branch2]
Target	[O][=C][Branch1][=C][N][C][=C][Branch1][C][F][C] [=C][C][=C][Ring1][#Branch1][F][C][=C][C][=C][Branch1] [Ring1][O][C][C][Branch1][C][O][=C][Ring1][=Branch2]
Levenshtein distance	**8**

Fig. 8 The Levenshtein distance between source and target SELFIES. The difference between the two SELFIES is highlighted in red colour

DRD2 datasets, which were 64.9% and 49.9%, respectively, our model achieved much lower collapse rates of 11.4% and 23.3%. To understand the collapse phenomenon better, we measured the number of collapses occurring at each Levenshtein distance between the generated and reconstructed SELFIES. The collapse frequency of STONED is considerably higher than that of our model for the same edit distance. In summary, the SELF-EdiT exhibited a comparatively lower collapse than STONED, demonstrating that the proposed method does not ignore the grammar of SELFIES for the editing task.

4.7 Edit path visualisation

SELF-EdiT exhibits explainability, as we can analyse the edit paths generated during the optimisation process to understand the specific structural modifications preferred by the model. Figure 10a shows an edit path with two iterations to improve DRD2. In the first iteration, SELF-EdiT identifies the base structure by removing unnecessary parts and adding a chain-like substructure. In the second iteration, no substructure was deleted, and the added chain part was refined. These edit path analyses may provide researchers with an opportunity to discover novel and vital substructures related to specific property optimisation. Figure 10b shows the simplified edit paths drawn using SELF-EdiT for the QED and DRD2 tasks. These results demonstrate that SELF-EdiT optimises molecular properties while retaining important substructures (e.g., scaffolds) of the source molecules.

5 Conclusion

In this study, we proposed SELF-EdiT, a SELFIES-based editing model, to efficiently optimise molecules under structural constraints by alternating between deletion and insertion operations. Our proposed model achieved a better performance on two widely used benchmark tasks. Furthermore,

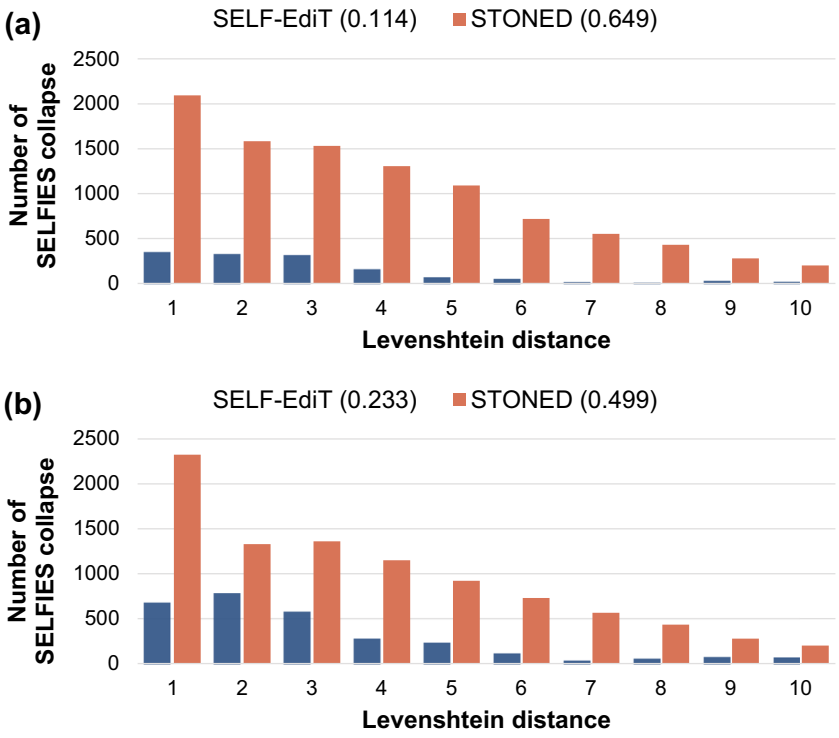


Fig. 9 Distributions of SELFIES collapse over Levenshtein distances. (a) QED and (b) DRD2. The x-axis and y-axis indicate the Levenshtein distance and the number of SELFIES collapse at each distance value, respectively

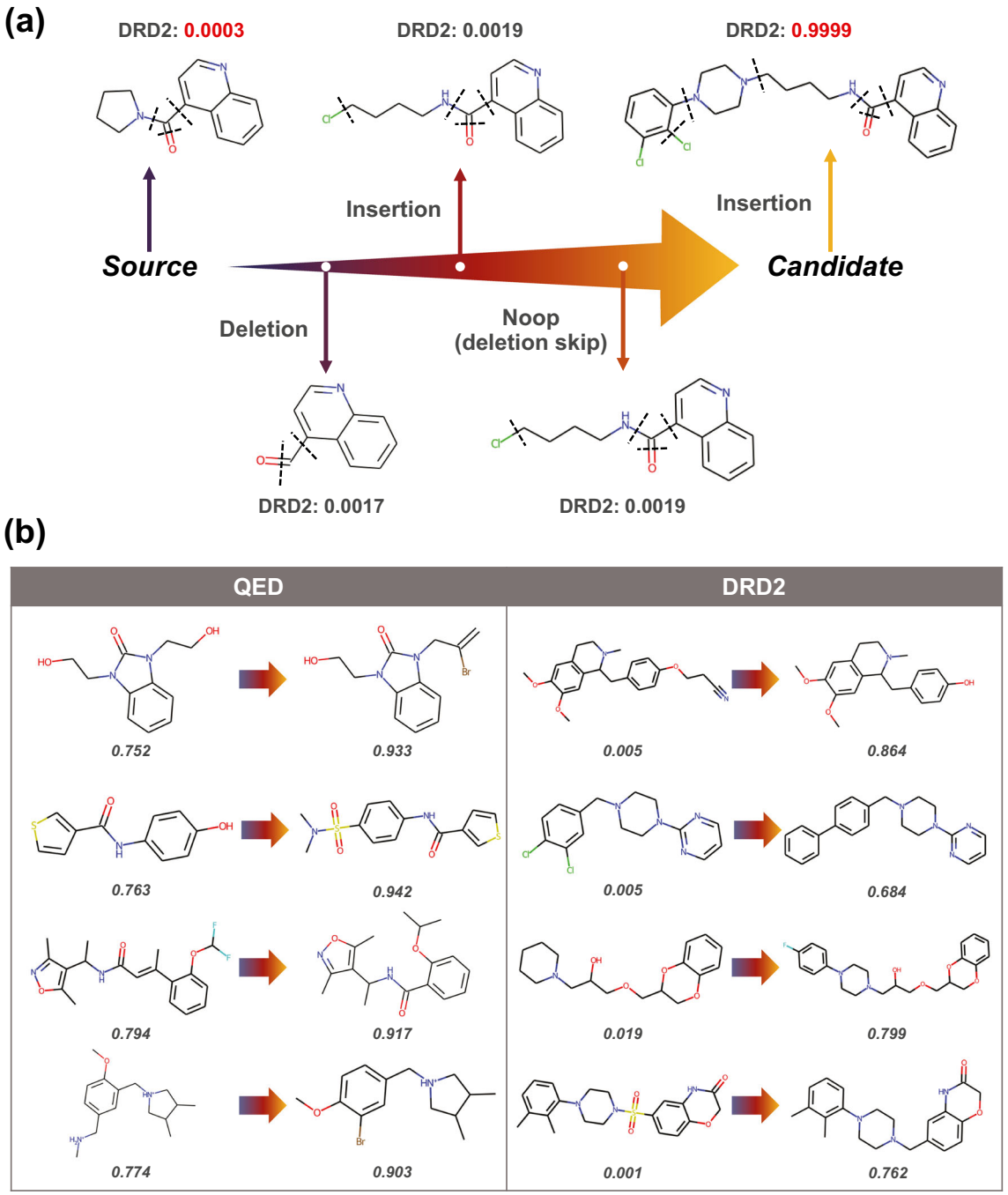


Fig. 10 Edit path visualisation. (a) Edit path of DRD2 optimisation with two edit iterations. The dashed lines represent the borderlines between SELFragments, (b) Simplified edit paths for QED and DRD2 optimisation

we confirmed that SELF-EdiT relieves the SELFIES collapse problem more effectively than the existing SELFIES-based models. We believe that our approach based on SELFIES editing offers a novel perspective on structure-constrained molecule optimisation, with potential applications in drug design and other related tasks. Although our proposed model showed promising results, there are limitations that need to be addressed in future research. One such limitation is that although we have demonstrated how SELF-EdiT edits molecules step-by-step through the editing path, the black box characteristic of the neural network makes it unclear how the model selects fragments at each editing operation. Therefore, future research should focus on developing a quantitative evaluation of the fragment decision to improve

the interpretability of the model, which is expected to make it a more reliable tool for chemical researchers to improve the efficiency of molecular optimisation.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10489-023-04915-8.

Funding This research was supported by the National Research Foundation (NRF) funded by the Korea government (MSIT) (NRF-2019R1A2C3005212 and RS-2023-00229822).

Data Availability The datasets generated during and/or analysed during the current study are available at https://github.com/sungmin630/SELF-EdiT.

Declarations

Conflicts of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Mullard A (2014) New drugs cost US $2.6 billion to develop. Nature Rev Drug Discov 13(12):877
2. Paul SM, Mytelka DS, Dunwiddie CT, Persinger CC, Munos BH, Lindborg SR, Schacht AL (2010) How to improve R&D productivity: the pharmaceutical industry's grand challenge. Nature Rev Drug Discov 9(3):203–214
3. Verdonk ML, Hartshorn MJ (2004) Structure-guided fragment screening for lead discovery. Curr Opin Drug Discov Dev 7(4):404–410
4. Gerry CJ, Schreiber SL (2018) Chemical probes and drug leads from advances in synthetic planning and methodology. Nature Rev Drug Discov 17(5):333–352
5. Polishchuk PG, Madzhidov TI, Varnek A (2013) Estimation of the size of drug-like chemical space based on GDB-17 data. J Comput-Aided Mol Des 27(8):675–679
6. Bilodeau C, Jin W, Jaakkola T, Barzilay R, Jensen KF (2022) Generative models for molecular discovery: Recent advances and challenges. Wiley Interdiscip Rev: Comput Mol Sci 12(5):1608
7. Yang S, Tan J, Chen B (2022) Robust spike-based continual meta-learning improved by restricted minimum error entropy criterion. Entropy 24(4):455
8. Yang S, Linares-Barranco B, Chen B (2022) Heterogeneous ensemble-based spike-driven few-shot online learning. Frontiers in Neuroscience 16
9. Yang S, Wang J, Zhang N, Deng B, Pang Y, Azghadi MR (2021) Cerebellumorphic: large-scale neuromorphic model and architecture for supervised motor learning. IEEE Trans Neural Netw Learn Syst 33(9):4398–4412
10. Yang S, Tan J, Lei T, Linares-Barranco B (2023) Smart traffic navigation system for fault-tolerant edge computing of internet of vehicle in intelligent transportation gateway. IEEE Transactions on Intelligent Transportation Systems
11. Weininger D (1988) SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. J Chem Inf Comput Sci 28(1):31–36
12. Krenn M, Häse F, Nigam A, Friederich P, Aspuru-Guzik A (2020) Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. Mach Learn: Sci Technol 1(4):045024
13. Deng J, Yang Z, Ojima I, Samaras D, Wang F (2022) Artificial intelligence in drug discovery: applications and techniques. Briefings in Bioinformatics 23(1)
14. Ertl P, Schuffenhauer A (2009) Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. J Cheminformatics 1:1–11
15. Yu J, Wang J, Zhao H, Gao J, Kang Y, Cao D, Wang Z, Hou T (2022) Organic compound synthetic accessibility prediction based on the graph attention mechanism. J Chem Inf Model 62(12):2973-2986
16. Jin W, Yang K, Barzilay R, Jaakkola T (2019) Learning multimodal graph-to-graph translation for molecular optimization. Paper presented at International Conference on Learning Representations 2019
17. Jin W, Barzilay R, Jaakkola T (2018) Junction tree variational autoencoder for molecular graph generation. In: International conference on machine learning, pp 2323–2332. PMLR
18. Jin W, Barzilay R, Jaakkola T (2020) Hierarchical generation of molecular graphs using structural motifs. In: International conference on machine learning, pp 4839–4848. PMLR
19. Ji C, Zheng Y, Wang R, Cai Y, Wu H (2021) Graph polish: A novel graph generation paradigm for molecular optimization. IEEE Transactions on Neural Networks and Learning Systems
20. Nigam A, Pollice R, Krenn M, dos Passos Gomes G, Aspuru-Guzik A (2021) Beyond generative models: superfast traversal, optimization, novelty, exploration and discovery (STONED) algorithm for molecules using SELFIES. Chem Sci 12(20):7079–7090
21. Gao W, Fu T, Sun J, Coley CW (2022) Sample efficiency matters: a benchmark for practical molecular optimization. Adv Neural Inf Process Syst 35:21342–21357
22. Kumar A, Voet A, Zhang KY (2012) Fragment based drug design: from experimental to computational approaches. Curr Med Chem 19(30):5128–5147
23. Gao T, Yao X, Chen D (2021) Simcse: Simple contrastive learning of sentence embeddings. In: Proceedings of the 2021 Conference on empirical methods in natural language processing, pp 6894–6910
24. Gu J, Wang C, Zhao J (2019) Levenshtein transformer. Advances in Neural Information Processing Systems 32
25. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser, Ł., Polosukhin I (2017) Attention is all you need. Advances in Neural Information Processing Systems 30
26. Levenshtein VI, et al (1966) Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, vol 10, pp 707–710. Soviet Union
27. You J, Liu B, Ying Z, Pande V, Leskovec J (2018) Graph convolutional policy network for goal-directed molecular graph generation. Advances in Neural Information Processing Systems 31

28. Zhou Z, Kearnes S, Li L, Zare RN, Riley P (2019) Optimization of molecules via deep reinforcement learning. Sci Rep 9(1):1–10
29. Bjorck J, Gomes CP, Weinberger KQ (2022) Is high variance unavoidable in rl? a case study in continuous control. Paper presented at International conference on learning representations 2022
30. Gómez-Bombarelli R, Wei JN, Duvenaud D, Hernández-Lobato JM, Sánchez-Lengeling B, Sheberla D, Aguilera-Iparraguirre J, Hirzel TD, Adams RP, Aspuru-Guzik A (2018) Automatic chemical design using a data-driven continuous representation of molecules. ACS Central Science 4(2):268–276
31. Griffiths R-R, Hernández-Lobato JM (2020) Constrained Bayesian optimization for automatic chemical design using variational autoencoders. Chem Sci 11(2):577–586
32. Moriconi R, Deisenroth MP, Sesh Kumar K (2020) High-dimensional Bayesian optimization using low-dimensional feature spaces. Mach Learn 109:1925–1943
33. Nigam A, Pollice R, Aspuru-Guzik A (2022) Parallel tempered genetic algorithm guided by deep neural networks for inverse molecular design. Digital Discov 1(4):390–404
34. Paszkowicz W (2009) Properties of a genetic algorithm equipped with a dynamic penalty function. Comput Mater Sci 45(1):77–83
35. Shuker SB, Hajduk PJ, Meadows RP, Fesik SW (1996) Discovering high-affinity ligands for proteins: SAR by NMR. Sci 274(5292):1531–1534
36. Murray CW, Rees DC (2009) The rise of fragment-based drug discovery. Nature Chem 1(3):187–192
37. Hadsell R, Chopra S, LeCun Y (2006) Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol 2, pp 1735–1742. IEEE
38. Olivecrona M, Blaschke T, Engkvist O, Chen H (2017) Molecular de-novo design through deep reinforcement learning. J Cheminformatics 9(1):1–14
39. Bickerton GR, Paolini GV, Besnard J, Muresan S, Hopkins AL (2012) Quantifying the chemical beauty of drugs. Nature Chem 4(2):90–98
40. Landrum G, et al (2013) RDKit: cheminformatics and machine learning software. RDKIT, ORG, p 405
41. Bajusz D, Rácz A, Héberger K (2015) Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? J Cheminformatics 7(1):1–13
42. Dalke A, Hert J, Kramer C (2018) mmpdb: An open-source matched molecular pair platform for large multiproperty data sets. J Chem Inf Model 58(5):902–910
43. Barshatski G, Radinsky K (2021) Unpaired generative molecule-to-molecule translation for lead optimization. In: Proceedings of the 27th ACM SIGKDD Conference on knowledge discovery & data mining, pp 2554–2564

Shengmin Piao received his B.S. in Computer Science & Technology from Yanbian University of Science and Technology, Yanji, China in 2020, and his M.S. in Computer Science from Yonsei University, Seoul, South Korea in 2022. He is currently pursuing his Ph.D. in Artificial Intelligence at Yonsei University, Seoul, South Korea. His current research interests include natural language reasoning and language generation.

Jonghwan Choi received his B.S. in Mathematics and M.S. in Computer Engineering from Incheon National University, Incheon, South Korea in 2016 and 2018, respectively, and his Ph.D. in Computer Science from Yonsei University, Seoul, South Korea in 2023. His current research interests include bioinformatics, cheminformatics, and AI-based drug design.

Sangmin Seo received his B.S. in Computer Engineering from the Incheon National University, Incheon, South Korea, and his M.S. in Computer Engineering from the Incheon National University, Incheon, South Korea. He is currently pursuing his Ph.D. in Computer Science at Yonsei University, Seoul, South Korea. His current research interests include machine learning and AI-based drug design.

Sanghyun Park received his B.S. and M.S. in Computer Engineering from Seoul National University, Seoul, South Korea in 1989 and 1991, respectively, and his Ph.D. in Computer Science from the University of California at Los Angeles, Los Angeles, California, USA in 2001. He is currently a Professor in the Department of Computer Science, Yonsei University, Seoul, South Korea. His current research interests include databases, data mining, bioinformatics, and big data system.