A Deep Generative Model for Molecule Optimization via One Fragment Modification

Ziqi Chen[1], Martin Renqiang Min[2], Srinivasan Parthasarathy[1,3], Xia Ning[1,3,4] ✉

[1]Computer Science and Engineering, The Ohio State University, Columbus, OH 43210. [2]Machine Learning Department, NEC Labs America, Princeton, NJ 08540. [3]Translational Data Analytics Institute, The Ohio State University, Columbus, OH 43210. [4]Biomedical Informatics, The Ohio State University, Columbus, OH 43210. ✉ning.104@osu.edu

Molecule optimization is a critical step in drug development to improve desired properties of drug candidates through chemical modification. We developed a novel deep generative model Modof over molecular graphs for molecule optimization. Modof modifies a given molecule through the prediction of a single site of disconnection at the molecule and the removal and/or addition of fragments at that site. A pipeline of multiple, identical Modof models is implemented into Modof-pipe to modify an input molecule at multiple disconnection sites. Here we show that Modof-pipe is able to retain major molecular scaffolds, allow controls over intermediate optimization steps and better constrain molecule similarities. Modof-pipe outperforms the state-of-the-art methods on benchmark datasets: without molecular similarity constraints, Modof-pipe achieves 81.2% improvement in octanol-water partition coefficient penalized by synthetic accessibility and ring size; and 51.2%, 25.6% and 9.2% improvement if the optimized molecules are at least 0.2, 0.4 and 0.6 similar to those before optimization, respectively. Modof-pipe is further enhanced into Modof-pipe^m to allow modifying one molecule to multiple optimized ones. Modof-pipe^m achieves additional performance improvement as at least 17.8% better than Modof-pipe.

Molecule optimization is a critical step in drug discovery to improve desired properties of drug candidates through chemical modification. For example, in lead (molecules showing both activity and selectivity towards a given target) optimization,[1] the chemical structures of the lead molecules can be altered to improve their selectivity and specificity. Conventionally, such molecule optimization process is planned based on knowledge and experiences from medicinal chemists, and is done via fragment-based screening or synthesis.[2–5] Thus, it is not scalable or automated. Recent *in silico* approaches using deep learning have enabled alternative computationally generative processes to accelerate the conventional paradigm. These deep-learning methods learn from string-based molecule representations (SMILES)[6,7] or molecular graphs,[8,9] and generate new ones accordingly (e.g., via connecting atoms and bonds) with better properties. While computationally attractive, these methods do not conform to the *in vitro* molecule optimization process in one very important aspect: molecule optimization needs to retain the major scaffold of a molecule, but generating entire, new molecular structures may not reproduce the scaffold. Therefore, these methods are limited in their potentials to inform and direct *in vitro* molecule optimization.

We propose a novel generative model for molecule optimization that better approximates *in silico* chemical modification. Our method is referred to modifier with one fragment, denoted as Modof. Following the idea of fragment-based drug design,[10,11] Modof predicts a single site of disconnection at a molecule, and modifies the molecule by changing the fragments (e.g., ring systems, linkers, side chains) at that site. Distinctly from existing molecule optimization approaches that encode and decode whole molecular graphs, Modof learns from and encodes the difference between molecules before and after optimization at one disconnection site. To modify a molecule, Modof generates only one fragment that instantiates the expected difference by decoding a sample drawn from the latent 'difference' space. Then, Modof removes the original fragment at the disconnection site, and attaches the generated fragment at the site. By sampling multiple times, Modof is able to generate multiple optimized candidates. A pipeline of multiple, identical Modof models, denoted as Modof-pipe, is implemented to optimize molecules at multiple disconnection sites through different Modof models iteratively, with the output molecule from one Modof model as the input to the next. Modof-pipe is further enhanced into Modof-pipe^m to allow modifying one molecule into multiple optimized ones as the final output.

Modof has the following advantages:

- Modof modifies one fragment at a time. It better approximates the *in vitro* chemical modification and retains the majority of molecular scaffolds. Thus, it potentially better informs and directs *in vitro* molecule optimization.
- Modof only encodes and decodes the fragment that needs modification and facilitates better modification performance.
- Modof-pipe modifies multiple fragments at different disconnection sties iteratively. It enables easier control over and intuitive deciphering of the intermediate modification steps, and facilitates better interpretability of the entire modification process.
- Modof is less complex compared to the state of the art (SOTA). It has at least 40% fewer parameters and uses 26% less training data.

- Modof-pipe outperforms the SOTA methods on benchmark datasets in optimizing octanol-water partition coefficient penalized by synthetic accessibility and ring size, with 81.2% improvement without molecular similarity constraints on the optimized molecules, and 51.2%, 25.6% and 9.2% improvement if the optimized molecules need to be at least 0.2, 0.4 and 0.6 similar (in Tanimoto over 2,048-dimension Morgan fingerprints with radius 2) to those before optimization, respectively.
- Modof-pipe^m further improves over Modof-pipe by at least 17.8%.
- Modof-pipe^m and Modof-pipe also show superior performance on two other benchmarking tasks optimizing molecule binding affinities against the dopamine D2 receptor, and improving the drug-likeness estimated by quantitative measures.

Related Work

A variety of deep generative models have been developed to generate molecules of desired properties. These generative models include reinforcement learning (RL)-based models, generative adversarial networks (GAN)-based models, flow-based generative models, and variational autoencoder (VAE)-based models, among others. Among RL-based models, You *et al.*[9] developed a graph convolutional policy network (GCPN) to sequentially add new atoms and corresponding bonds to construct new molecules. In the flow-based models, Shi *et al.*[12] developed an autoregressive model (GraphAF), in which they learned an invertible mapping between Gaussian distribution and molecule structures, and applied RL to fine tune the generation process. Zang and Wang[13] developed a flow-based method (MoFlow), in which they utilized bond flow to learn an invertible mapping between bond adjacency tensors and Gaussian distribution, and then applied a graph conditional flow to generate an atom-type matrix given the bond adjacency tensors. Variational autoencoder (VAE)-based generative models are also very popular in molecular graph generation. Jin *et al.*[8] first decomposed a molecular graph into a junction tree of chemical substructures, and then used a junction tree VAE (JT-VAE) to generate and assemble new molecules. Jin *et al.*[14] developed a junction tree-based encoder-decoder neural model (JTNN), which learns a translation mapping between a pair of molecules to optimize one into another. Jin *et al.*[15] replaced the small chemical substructures used in JT-VAE with larger graph motifs, and modified JTNN into an autoregressive hierarchical encoder-decoder model (HierG2G). Additional related work including fragment-based VAE,[16] Teacher and Student polish (T&S polish),[17] scaffold-based VAE[18] and other genetic algorithm-based methods[19,20] are discussed in Section S1[1].

The existing generative methods typically encode the entire molecular graphs, and generate whole, new molecules from an empty or a randomly selected structure. Different from these methods, Modof learns from and encodes the *difference* between molecules before and after optimization. Thus, the learning and generative processes is less complex, and are able to retain major molecular scaffolds.

Problem Definition

Following Jin *et al.*,[8] we focus on the optimization of the partition coefficients (logP) measured by Crippen logP[21] and penalized by synthetic accessibility[22] and ring size. Crippen logP is a predicted value of experimental logP using the Wildman and Crippen approach,[21] and has been demonstrated to have a strong correlation (e.g., r^2=0.918[21]) with experimental logP. Since it is impractical to measure the experimental logP values for a large set of molecules, such as our training set (Section S3), or for *in silico* generated molecules, using Crippen logP will enable the scalable learning from a large set of molecules, and effective yet accurate evaluation on *in silico* optimized molecules. The combined measurement of logP, synthetic accessibility (SA) and ring size is referred to as *penalized* logP, denoted as plogP. Higher plogP values indicate higher molecule concentrations in the lipid phase with potentially good synthetic accessibility and simple ring structures. Note that Modof can be used to optimize other properties as well, with the property of interest used instead of plogP. Optimizing other properties is discussed in the Section S11. Optimizing multiple properties simultaneously is discussed in the Section S12. In the rest of this document, "property" is by default referred to plogP.

Problem Definition: Given a molecule M_x, molecule optimization aims to modify M_x into another molecule M_y such that 1) M_y is similar to M_x in its molecular structures (similarity constraint), that is, $\text{sim}(M_x, M_y) \geq \delta$ (δ is a threshold); and 2) M_y is than better than M_x in the property of interest (e.g., $\text{plogP}(M_y) > \text{plogP}(M_x)$) (property constraint).

Materials

Data

We used the benchmark training dataset provided by Jin *et al.*.[15] This dataset was extracted from ZINC dataset[23,24] and contains 75K pairs of molecules. Every two paired molecules are similar in their molecule structures but different in their plogP values. Using DF-GED[25] algorithm, we then extracted 55,686 pairs of molecules from Jin's training dataset such that each extracted pair has only one disconnection site. That is, our training data is 26% less than that in Jin's. We used these extracted pairs of molecules (104,708 unique molecules) as our training data. Details about the training data generation are discussed as follows. We used Jin's validation set for parameter tuning and tested on Jin's test dataset of 800 molecules. More details about the training data are available in Section S3.

[1]Section references starting with "S" refer to a Supplementary Information Section.

Training Data Generation

We used a pair of molecules (M_x, M_y) as a training instance in Modof, where M_x and M_y satisfy both the similarity and property constraints, and M_y is different from M_x in only one fragment at one disconnection site. We constructed such training instances as follows. We first quantified the difference between M_x and M_y using the optimal graph edit distance[26] between their junction tree representations $\mathcal{T}_x$ and $\mathcal{T}_y$, and derived the optimal edit paths to transform $\mathcal{T}_x$ to $\mathcal{T}_y$. Such quantification also identified disconnection sites at M_x during its graph comparison. Details about this process is available in Section S4. Identified molecule pairs satisfying similarity and property constraints with only one site of disconnection were used as training instances. For a pair of molecules with a high similarity (e.g., above 0.6), it is very likely that they have only one disconnection site as demonstrated in Section S5.

Molecule Similarity Calculation

We used 2,048-dimension binary Morgan fingerprints with radius 2 to represent molecules, and used Tanimoto coefficient to measure molecule similarities.

Baseline Methods

We compared Modof with the state-of-the-art baseline methods for the molecule optimization, including JT-VAE,[8] GCPN,[9] JTNN,[14] HierG2G,[15] GraphAF[12] and MoFlow.[13]

- JT-VAE encodes and decodes junction trees, and assembles new, entire molecular graphs based on decoded junction trees.
- GCPN applies a graph convolutional policy network and iteratively generates molecules by adding atoms and bonds one by one.
- JTNN learns from molecule pairs and performs molecule optimization as to translate molecular graphs.
- HierG2G encodes molecular graphs in a hierarchical fashion, and generates new molecules via generating and connecting structural motifs.
- GraphAF learns an invertible mapping between a prior distribution and molecular structures, and uses reinforcement learning to fine-tune the model for molecule optimization.
- MoFlow learns an invertible mapping between bond adjacency tensors and Gaussian distribution, and then applies a graph conditional flow to generate an atom-type matrix as the representation of a new molecule from the mapping.

Experimental Results

Overall Comparison on plogP Optimization

Table 1 | Overall Comparison on Optimizing plogP

model	$\delta = 0.0$		$\delta = 0.2$		$\delta = 0.4$		$\delta = 0.6$	
	imprv±std	sim±std	imprv±std	sim±std	imprv±std	sim±std	imprv±std	sim±std
JT-VAE	1.91±2.04	0.28±0.15	1.68±1.85	0.33±0.13	0.84±1.45	0.51±0.10	0.21±0.71	0.69±0.06
GCPN	4.20±1.28	0.32±0.12	4.12±1.19	0.34±0.11	2.49±1.30	0.47±0.08	0.79±0.63	0.68±0.08
JTNN	-	-	-	-	3.55±1.54	0.46±0.06	2.33±1.19	0.66±0.05
HierG2G	-	-	-	-	3.98±1.46	0.46±0.06	2.49±1.09	0.66±0.05
GraphAF	2.94±1.55	0.31±0.15	2.65±1.29	0.35±0.12	1.62±1.16	0.51±0.10	0.34±0.46	0.69±0.06
MoFlow	2.39±1.47	0.54±0.22	2.26±1.37	0.59±0.17	2.04±1.24	0.65±0.12	1.46±1.09	0.71±0.07
Modof-pipe	7.61±2.30	0.21±0.15	6.23±1.77	0.34±0.12	5.00±1.53	0.48±0.09	2.72±1.53	0.65±0.05
Modof-pipe^m	9.37±2.04	0.12±0.08	7.58±1.65	0.27±0.07	5.89±1.57	0.46±0.06	3.14±1.77	0.65±0.05

Columns represent: "imprv": the average improvement in plogP; "std": the standard deviation; "sim": the similarity between the original molecules M_x and optimized molecules M_y; "-": not reported in literature. We calculated "sim±std" for JTNN and HierG2G using the optimized molecules provided by JTNN and our reproduced results for HierG2G, respectively.

Table 1 presents the overall comparison among Modof-pipe and Modof-pipe^m, both with a maximum of 5 iterations, and the baseline methods on plogP optimization. Note that Modof-pipe^m outputs 20 optimized molecules as JTNN and HierG2G do. Following GCPN, an additional constraint of molecule size is imposed into Modof-pipe to limit the size of optimized molecules to be at most 38. As Crippen logP tends to be large on large molecules, this additional constraint also prevents Modof-pipe from improving logP by simply increasing molecule size. When there is no similarity constraint (δ=0), that is, it is not required to produce similar molecules out of the optimization, Modof-pipe is able to generate highly optimized molecules with substantially better plogP improvement (7.61±2.30), with 81.2% improvement from the best baseline GCPN (4.20±1.28), although with lower similarities between the molecules before and after the optimization. Modof-pipe^m achieves even better performance with plogP improvement 9.37±2.04, that is, 123.1% better than GCPN. When the similarity constraint takes effect (e.g., δ=0.2, 0.4 and 0.6), Modof-pipe consistently produces molecules that are both similar to those before optimization and also with better properties. At δ=0.2, 0.4 and 0.6, Modof-pipe achieves better property improvement (6.23±1.77, 5.00±1.53 and 2.72±1.68, respectively) than all the best baselines (GCPN with 4.12±1.19 at δ=0.2, HierG2G with 3.98±1.47 at δ=0.4 and 2.49±1.09 at δ=0.6), with 51.2%, 25.6% and 9.2% improvement over the baselines, respectively, though the baselines generate more similar molecules than Modof-pipe; Modof-pipe^m achieves the best performance on property improvement (7.58±1.65, 5.89±1.57, 3.14±1.77, respectively) with 84.0%, 48.0% and 26.1% improvement over the best baselines, respectively.

When δ is large, we could observe that JTNN and HierG2G tend to decode more aromatic rings, leading to large molecules with over-estimated similarities. Instead, Modof tends to stop if there are many aromatic rings, and thus, produces more drug-like molecules.[27,28] Issues related to similarity calculation that will affect optimization performance are discussed in Section S7. Still, the overall comparison demonstrates that Modof-pipe and Modof-pipe^m outperform or at least achieve similar performance as the state-of-the-art methods.

It is worth noting that our performance is reported on the exact benchmark test set. In our study, we observed some issues of unfair comparison in the existing baseline methods. For example, some baseline methods compared and reported results on a different test set rather than the benchmark test set. Some reinforcement learning methods used the test molecules to either directly train a model or fine-tune a pre-trained model to optimize the test molecules, which may lead to artificially high performance.[29,30] Detailed discussions on comparison fairness are available in Section S8.

Additional experimental results are available in Section S9, such as overall Modof-pipe performance, transformation over chemical spaces, and retaining of molecule scaffolds. Specifically, we compared model complexities (Section S9.7), which shows that Modof uses at least 40% [illegible] training data but outperforms or achieves very cc
Se

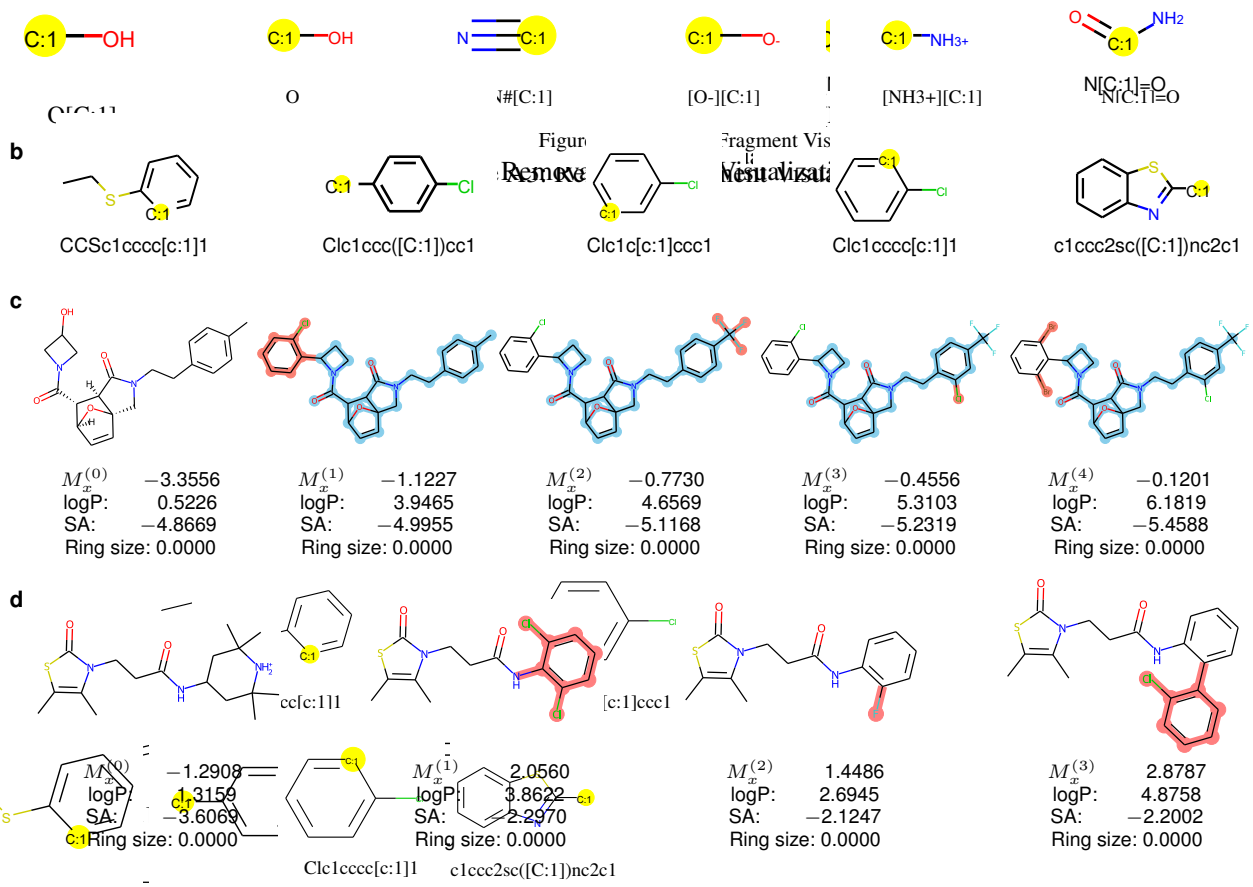


Fig. 1c Modof-pipe Examples for plogP Optimization. Visualization of popular removal fragments. **b,** Visualization of popular attaching fragments. **c,** Modof-pipe optimization example with multiple disconnection sites and multiple Modof iterations. **d,** Local optimization.

Case Study

Among training molecules, the top-5 most popular fragments that have been removed from M_x are presented in Fig. 1a with their canonical SMILE strings; the top-5 most popular fragments to be attached to generate M_y are presented in Fig. 1b. Overall, the removal fragments from training data are on average of 2.85 atoms and the new attached fragments are of 7.55 atoms, that is, the optimization is typically done via removing small fragments and then attaching larger fragments.

Fig. 1c presents an example of molecule M_x (i.e., $M_x^{(0)}$) being optimized via four iterations in Modof-pipe into another molecule $M_x^{(4)}$ under δ=0.4. At each iteration, only one, small fragment (highlighted in red in the figure) is modified from its input, and plogP value (below each molecule) is improved. In the first iteration, $M_x^{(1)}$ is modified from $M_x^{(0)}$ via the removal of the hydroxyl group in $M_x^{(0)}$ and the addition of the 2-chlorophenyl group. The hydroxyl group is polar and tends to increase water solubility of the molecules, while the 2-chlorophenyl group is non-polar and thus more hydrophobic. In addition, the increase in molecular weight brought by the chlorophenyl substituent would contribute to the lower water solubility as well. Thus, the modification from the hydroxyl group to the chlorophenyl group induces the logP increase (from 0.5226 to 3.9465). Meanwhile, the introduction of the 2-chlorophenyl group to the cyclobutyl group adds complexity to the synthesis, in addition to possible steric effects due to the *ortho*-substitution on the aromatic ring, and induces a decrease in synthetic accessibility (SA) (from −4.8669 to −4.9955). In the second iteration, the methyl

group in $M_x^{(1)}$ is replaced by a trifluoromethyl group. The trifluoromethyl group is more hydrophobic than the methyl group, and thus increases the logP value of $M_x^{(2)}$ over $M_x^{(1)}$ (from 3.9465 to 4.6569). Meanwhile, the slightly larger molecule $M_x^{(2)}$ has slightly worse SA (from −4.9955 to −5.1168). If logP is preferred to be lower than 5 as proposed in the Lipinski's Rule of Five,[31] Modof-pipe can be stopped at this iteration; otherwise, in the following two iterations, more halogens are added to the aromatic ring, which could make the aromatic ring less polar and further decrease water solubility and increase logP values.[32] These four iterations highlight the interpretability of Modof-pipe corresponding to chemical knowledge. Please note that all the modifications in Modof are learned in an end-to-end fashion from data without any chemical rules or templates imposed *a priori*, emphasizing the power of Modof in learning from molecules.

In Fig. 1c, the molecule similarities between $M_x^{(t)}$ (t=1,..,4) and $M_x^{(0)}$ are 0.630, 0.506, 0.421, 0.4111, respectively. This example also shows that Modof is able to retain the major scaffold of a molecule and optimizes at different disconnection sites during the iterative optimization process. Additional analysis on fragments is available in Section S10.

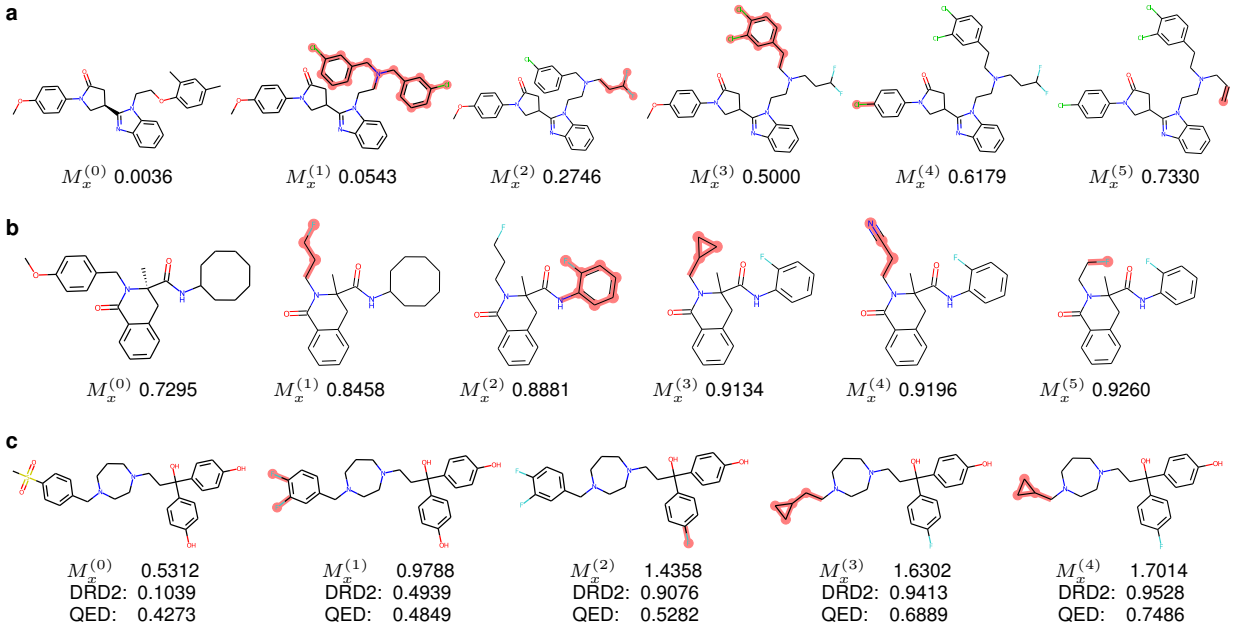


Fig. 2 | Modof-pipe examples for DRD2, QED and multi-property optimization. a, Modof-pipe examples for DRD2 optimization. **b,** Modof-pipe examples for QED optimization. **c,** Modof-pipe examples for multi-property optimization of DRD2 and QED.

Table 2 | Overall Comparison on Optimizing DRD2 and QED

model	Optimizing DRD2						Optimizing QED					
	OM-pic (DRD2(M_y) $\geq$ 0.5)			OM-trn (imprv $\geq$ 0.2)			OM-pic (QED(M_y) $\geq$0.9)			OM-trn (imprv$\geq$0.1)		
	rate%	imprv±std	sim±std	rate%	imprv±std	sim±std	rate%	imprv±std	sim±std	rate%	imprv±std	sim±std
JTNN	78.10	0.83±0.17	0.44±0.05	78.30	0.83±0.17	0.44±0.05	60.50	0.17±0.03	0.47±0.06	67.38	0.17±0.03	0.47±0.07
HierG2G	82.00	0.83±0.16	0.44±0.05	84.00	0.82±0.18	0.44±0.05	**75.12**	0.18±0.03	0.46±0.06	82.38	0.17±0.03	0.46±0.06
JTNN(m)	43.50	0.77±0.15	0.49±0.08	61.60	0.65±0.24	0.49±0.08	40.50	0.17±0.03	0.54±0.09	68.50	0.15±0.03	0.54±0.09
HierG2G(m)	51.80	0.78±0.15	0.49±0.08	70.20	0.66±0.24	0.49±0.08	37.12	0.17±0.03	0.52±0.09	65.88	0.15±0.03	0.53±0.10
Modof-pipe	74.90	0.83±0.14	0.48±0.07	89.00	0.75±0.22	0.48±0.07	40.00	0.17±0.03	0.51±0.08	70.00	0.16±0.03	0.51±0.08
Modof-pipe^m	**88.60**	0.88±0.12	0.46±0.05	**95.90**	0.84±0.18	0.46±0.05	66.25	0.18±0.03	0.48±0.07	**87.62**	0.17±0.03	0.48±0.07

Columns represent: OM-pic: the optimized molecules that achieve a certain property improvement: (1) for DRD2, the optimized molecules M_y should have DRD2 score no less than 0.5; (2) for QED, the optimized molecules M_y should have QED score no less than 0.9. OM-trn: the optimized molecules that achieve a property improvement in a similar degree as in training data: (1) for DRD2, the optimized molecules M_y should satisfy DRD2(M_y) − DRD2(M_x) $\geq$ 0.2; (2) for QED, the optimized molecules M_y should satisfy QED scores QED(M_y) − QED(M_x) $\geq$ 0.1. "rate%": the percentage of optimized molecules in each group (OM, OM-pic, OM-trn) over all test molecules; "imprv": the average property improvement; "std": the standard deviation; "sim": the similarity between the original molecules M_x and optimized molecules M_y. Best rate% values are in **bold.**

Performance on DRD2 and QED Optimization

In addition to improving plogP, another two popular benchmarking tasks for molecule optimization include improving molecule binding affinities against the dopamine D2 receptor (DRD2), and improving the drug-likeness estimated by quantitative measures (QED).[33] Specifically, given a molecule that doesn't bind well to the DRD2 receptor (e.g., with low binding affinities), the objective of optimizing DRD2 property is to modify the molecule into another one that will better bind to DRD2. In the QED task, given a molecule that is not much drug-like, the objective of optimizing QED property is to modify this molecule into a more "drug-like" molecule. Table 2 presents the major results in success rates, property improvement and similarity comparison under the similarity constraint δ=0.4. The results demonstrate that Modof-pipe^m significantly outperforms or is comparable to the baseline methods in optimizing DRD2 and QED, when the success rates are measured using either the benchmark metrics[14,15] (OM-pic in Table 2) or based on training data (OM-trn in Table 2). Fig. 2a and Fig. 2b present two examples of molecule optimization for DRD2 and QED property

improvement. Particularly, as in Fig. 2b, in the first iteration, a 4-methoxyphenyl group is removed and a small chain of 2-fluoroethyl group is added, and thus, the number of aromatic rings and the number of hydrogen bond acceptors are reduced, which makes the compound more drug-like than its predecessor. In the second iteration, a cyclooctyl group is removed from ${M_x}^{(1)}$ and a 2-fluorophenyl group is added. This modification may induce reduced flexibility – another preferred property of a successful drug. In the following iterations, some commonly used fragments in drug design are used to further modify the molecule into more drug-likeness. Note that, again, QED optimization is completely learned from data in an end-to-end fashion without any medicinal chemistry knowledge imposed by experts. The meaningful optimization in the example in Fig. 2b demonstrates the interpretability of Modof-pipe. More details about these two optimization tasks and results are available in the Section S11.

We also conducted experiments to optimize both DRD2 and QED properties of molecules simultaneously, that is, a multi-property optimization task. Details on this multi-property task and results are available in Section S12. Fig. 2c presents an example of multi-property molecule optimization, in which both the DRD2 and QED scores of the molecule are consistently increased with the iterations of optimization.

Discussions and Conclusions

Molecule Optimization using Simulated Properties

Most of the molecule properties considered in our experiments are based on simulated or predicted values rather than experimentally measured. That is, an independent simulation or machine learning model is first used to generate the property values for the benchmark dataset. For example, Crippen logP is estimated via the Wildman and Crippen approach;[21] synthesis accessibility is calculated using a scoring function over predefined fragments;[22] the DRD2 property is predicted using a support vector machine classifier;[34] and the QED property is predicted using a non-linear classifier combining multiple desirability functions of molecular properties.[33] While all the existing generative models for molecule optimization[8,9,13–16,19,35–37] use such simulated properties, there are both challenges and opportunities. Challenges arise when the simulation or machine learning models for those property predictions are not sufficiently accurate due to various reasons (e.g., limited or biased training molecules), the generative models learned from the inaccurate property values would also be inaccurate or incorrect, resulting in generated molecules that could negatively impact the downstream drug development tasks significantly. However, the opportunities due to the property simulation or prediction can be immense in fully unleashing the power of large-scale, data-driven learning paradigms to stimulate drug development as we continue to improve these simulations and predictions. Specifically, most deep learning-based models for drug development purposes, many of which have been demonstrated to be very promising,[38] are not possible without large-scale training data. While it is impractical, if ever possible, to experimentally measure the interested properties for a large set of molecules (e.g., more than 100K molecules as in our benchmark training data), the property simulation or prediction of the molecules enables large training data and makes the development of such deep learning methodologies possible. Fortunately, property prediction simulations or models have become more accurate (e.g., 98% accuracy for DRD2[34]) due to the accumulation of experimental measurements[39] and the strong learning power of innovative computational approaches. The accurate property simulation or prediction over large-scale molecule data and the powerful learning capability of generative models from such molecule data will together have strong potentials to further advance *in silico* drug development.

Synthesizability and Retrosynthesis

Our experiments show that Modof is also able to improve synthesis accessibility (Section S9.4). However, it does not necessarily mean that the generated molecules can be easily synthesized. This limitation of Modof is actually common for almost all the computational approaches for molecule generation. A recent study shows that many molecules generated via deep learning are not easily synthesizable,[40] which significantly limits the translational potentials of the generative models in making real impacts in drug development. On the other hand, retrosynthesis prediction via deep learning, which aims to identify a feasible synthesis path for a given molecule through learning and searching from a large collection of synthesis paths, has been an active research area.[41,42] Optimizing molecules towards not only better properties but also better synthesizability, particularly with explicit synthesis paths identified simultaneously, could be a highly interesting and challenging future research direction. Ultimately, we would like to develop a comprehensive computational framework that could generate synthesizable molecules with preferable properties. This would require not only a substantial amount of data to train sophisticated models, but also necessary domain knowledge and human experts looped in the learning process.

In vitro Validation

Testing the *in silico* generated molecules in a laboratory will be needed ultimately to validate the computational methods. While currently most existing computational methods are developed in academic environments and thus cannot be easily tested on purchasable or proprietary molecule libraries, or cannot be easily synthesized as we discussed earlier, a few successful stories[43] have demonstrated that powerful computational methods have high potentials to truly make new discoveries that can succeed in laboratory validation. Analogous to this molecule optimization and discovery process using deep learning approaches is from AlphaFold,[44] a deep learning method predicting protein folding structures. The breakthrough from AlphaFold in solving a 50-year-old grand challenge in biology offers a strong evidence showing the tremendous power of modern learning approaches, which should not be underestimated. Still, collaborations with pharmaceutical industry and *in vitro* test are highly needed to truly translate the computational methods into real impact. In addition, effective sampling and/or prioritization of generated molecules in order to identify a feasible, small set of

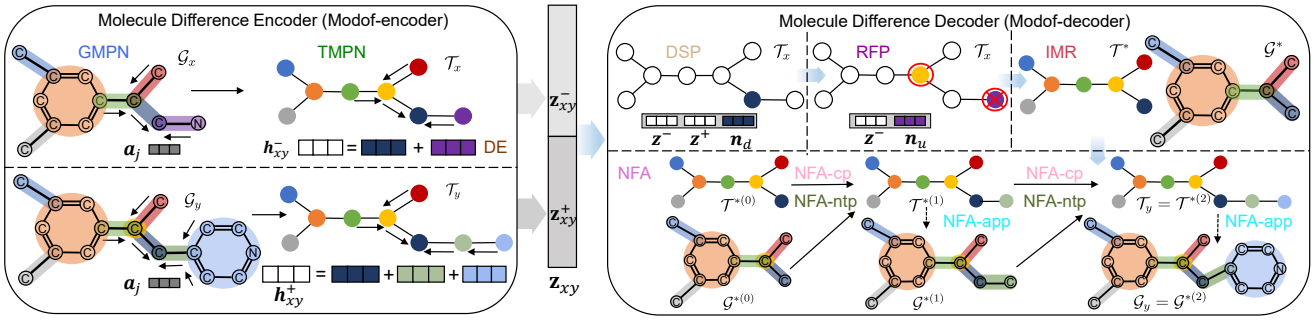
molecules for small-scale *in vitro* validation could be a practical solution; it will require the development of new sampling schemes over molecule subspaces, and/or the learning of molecule prioritization[45,46] within the molecule generation process. Meanwhile, large-scale *in vitro* validation of *in silico* generated molecules represents a challenging but interesting future research direction.

Other Issues in Computational Molecule Optimization

A limitation of Modof-pipe is that it employs a local greedy optimization strategy: in each iteration, the input molecules to Modof will be optimized to the best, and if the optimized molecules do not have better properties, they will not go through additional Modof iterations. Detailed discussions on local greedy optimization are available in Section S13.1. In addition to partition coefficient, there are a lot of factors (e.g., toxicity, synthesizability) that need to be considered in order to develop a molecule into a drug. Discussions on multi-property optimization are available in Section S13.2. Target-specific molecule optimization are also discussed in Section S13.3. The Modof framework could also be used for compounds or substance property optimization in other application areas (e.g., melting or boiling points for volatiles). Related discussions are available in Section S13.4.

Conclusions

Modof optimizes molecules at one disconnection site at a time by learning the difference between molecules before and after optimization. With a much less complex model, it achieves significantly better or similar performance compared to the states of the art. In addition to the limitations and corresponding future research directions that have been discussed above, another limitation with Modof is that in Modof, the modification happens at the periphery of molecules. Although this is very common in *in vitro* lead optimization, we are currently investigating how Modof can be enhanced to modify the internal regions of molecules, if needed, by learning from proper training data with such regions. Additionally, we hope to integrate domain-specific knowledge in the Modof learning process facilitating increased explainability in the learning and generative process.



Modof-encoder: Modof first generates atom embeddings of M_x/M_y over molecular graphs $\mathcal{G}_x/\mathcal{G}_y$ using graph message passing networks (GMPN), and node embeddings over corresponding junction tree $\mathcal{T}_x/\mathcal{T}_y$ using tree message passing networks (TMPN). The difference between $\mathcal{T}_x$ and $\mathcal{T}_y$ at the disconnection site ($\bullet$ in $\mathcal{T}_x/\mathcal{T}_y$) is encoded (DE) into $\mathbf{h}^-_{xy}$ and $\mathbf{h}^+_{xy}$, which then construct two normal distributions $\mathbf{z}^-_{xy}$ and $\mathbf{z}^+_{xy}$.
Modof-decoder: Using $\mathbf{z}_{xy}$, Modof conducts disconnection site prediction (DSP) to identify the site n_d. At neighbors of n_d, Modof conducts removal fragment prediction (RFP) to remove fragment at n_d. Then, Modof produces an intermediate representation (IMR) of the remaining scaffold ($\mathcal{G}^*$, $\mathcal{T}^*$). Over ($\mathcal{G}^*$, $\mathcal{T}^*$), Modof performs new fragment attachment (NFA) by interactively performing child node connection prediction (NFA-cp), child node type prediction (NFA-ntp), and attachment point prediction (NFA-app) to optimize M_x.
Molecule representations: substructures in molecular graphs and their corresponding nodes in junction trees are coded in a same color.

Fig. 3 | Modof Model Overview.

Methods

Modof modifies one fragment (e.g., a ring system, a linker, a side chain) of a molecule at a time, and thus only encodes and decodes the fragment that needs modification. The site of M where the fragment is modified is referred to as the site of disconnection and denoted as n_d, which corresponds to a node in the junction tree representation (discussed in "Molecule Representations and Notations"). Fig. 3 presents an overview of Modof. All the algorithms are presented in Section S14. Discussions on the single-disconnection-site rationale are presented in Section S5.

Molecule Representations and Notations

We represent a molecule M_x using a molecular graph $\mathcal{G}_x$ and a junction tree $\mathcal{T}_x$. $\mathcal{G}_x$ is denoted as $\mathcal{G}_x = (\mathcal{A}_x, \mathcal{B}_x)$, where $\mathcal{A}_x$ is the set of atoms in M_x, and $\mathcal{B}_x$ is the set of corresponding bonds. In the junction tree representation $\mathcal{T}_x = (\mathcal{V}_x, \mathcal{E}_x)$,[8] all the rings and bonds in M_x are extracted as nodes in $\mathcal{V}_x$; nodes with common atoms are connected with edges in $\mathcal{E}_x$. Thus, each node $n \in \mathcal{V}_x$ is a substructure (e.g., a ring, a bond and its connected atoms) in $\mathcal{G}_x$. We denote the atoms included in node n as $\mathcal{A}_x(n)$ and refer to the nodes connected to n in $\mathcal{T}_x$ as its neighbors, denoted as $\mathcal{N}_x(n)$. Thus, each edge $(n_u, n_v) \in \mathcal{E}_x$ actually corresponds to the common atoms $\mathcal{A}_x(n_u) \cap \mathcal{A}_x(n_v)$ between n_u and n_v. When no ambiguity arises, we will eliminate subscript x in the notations. Note that atoms and bonds are the terms used for molecular graph representations, and nodes and edges are used for junction tree representations. In this manuscript, all the embedding vectors are by default column vectors, represented by lower-case bold letters; all the matrices are represented by upper-case letters. Key notations are listed in Table 3.

Table 3 | Notations

notation	meaning
$M = (\mathcal{G}, \mathcal{T})$	molecule represented by $\mathcal{G}$ and $\mathcal{T}$
$\mathcal{G} = (\mathcal{A}, \mathcal{B})$	molecular graph with atoms $\mathcal{A}$ and bonds $\mathcal{B}$
$\mathcal{T} = (\mathcal{V}, \mathcal{E})$	junction tree with nodes $\mathcal{V}$ and edges $\mathcal{E}$
a	an atom in $\mathcal{G}$
b_{ij}	a bond connecting atoms a_i and a_j in $\mathcal{G}$
n	a node in $\mathcal{T}$
e_{uv}	an edge connecting nodes n_u and n_v in $\mathcal{T}$
n_d	site of disconnection
$\mathcal{A}(n), \mathcal{N}(n)$	atoms included in a tree node n, n's neighbors
$\mathbf{x}$	atom type embedding
$\mathbf{m}^{(1\cdots t)}$	concatenation of $\mathbf{m}^{(1)}$, $\mathbf{m}^{(2)}$, ... $\mathbf{m}^{(t)}$

Molecular Difference Encoder (Modof-encoder)

Given two molecules (M_x, M_y), Modof (Algorithm S1 in Section S14) learns and encodes the difference between M_x and M_y using message passing networks[47] over graphs $\mathcal{G}_x$ and $\mathcal{G}_y$, denoted as GMPN, and over junction trees $\mathcal{T}_x$ and $\mathcal{T}_y$, denoted as TMPN, via three steps.

Step 1. Atom Embedding over Graphs (GMPN)

Modof first represents atoms using embeddings to capture atom types and their local neighborhood structures by propagating messages along bonds over molecular graphs. Modof uses an one-hot encoding $\mathbf{x}_i$ to represent the type of atom a_i, and an one-hot encoding $\mathbf{x}_{ij}$ to represent the type of bond b_{ij} connecting a_i and a_j. Each bond b_{ij} is associated with two messages $\mathbf{m}_{ij}$ and $\mathbf{m}_{ji}$ encoding the messages propagating from atom a_i to a_j and vice versa. The $\mathbf{m}_{ij}^{(t)}$ in t-th iteration of GMPN is updated as follows:

$$\mathbf{m}_{ij}^{(t)} = \mathrm{ReLU}(W_1^a \mathbf{x}_i + W_2^a \mathbf{x}_{ij} + W_3^a \sum_{a_k \in \mathcal{N}(a_i) \setminus \{a_j\}} \mathbf{m}_{ki}^{(t-1)}),$$

where $\mathbf{m}_{ki}^{(0)}$ is initialized as zero, and W_i^a's (i=1,2,3) are the learnable parameter matrices. Thus, the message $\mathbf{m}_{ij}^{(t)}$ encodes the information of all length-t paths passing through b_{ij} to a_j in the graph. After t_a iterations of message passing, the atom embedding $\mathbf{a}_j$ is updated as follows:

$$\mathbf{a}_j = \mathrm{ReLU}(U_1^a \mathbf{x}_j + U_2^a \sum_{a_i \in \mathcal{N}(a_j)} \mathbf{m}_{ij}^{(1\cdots t_a)}),$$

where $\mathbf{m}_{ij}^{(1\cdots t_a)}$ is the concatenation of message vectors from all iterations, and U_1^a and U_2^a are learnable parameter matrices. Thus, the atom embedding $\mathbf{a}_j$ aggregates information from a_j's t_a-hop neighbors, similarly to Xu *et al.*,[48] to improve the atom embedding representation power.

Step 2. Node Embedding over Junction Trees (TMPN)

Modof encodes nodes in junction trees into embeddings to capture their local neighborhood structures by passing messages along the tree edges. To produce rich representations of nodes, Modof first aggregates the information of atoms within a node n_u into an embedding $\mathbf{s}_u$, and the information of atoms shared by a tree edge e_{uv} into an embedding $\mathbf{s}_{uv}$ through the following pooling:

$$\mathbf{s}_u = \sum_{a_i \in \mathcal{A}(n_u)} \mathbf{a}_i, \quad (1) \qquad \mathbf{s}_{uv} = \sum_{a_i \in \mathcal{A}(n_u) \cap \mathcal{A}(n_v)} \mathbf{a}_i. \quad (2)$$

Modof also uses a learnable embedding $\mathbf{x}_u$ to represent the type of node n_u. Thus, $\mathbf{m}_{uv}^{(t)}$ from node n_u to n_v in t-th iteration of TMPN is updated as follows:

$$\mathbf{m}_{uv}^{(t)} = \mathrm{ReLU}(W_1^n \mathrm{ReLU}(W_2^n [\mathbf{x}_u; \mathbf{s}_u]) + W_3^n \mathbf{s}_{uv} + W_4^n \sum_{n_w \in \mathcal{N}(n_u) \setminus \{n_v\}} \mathbf{m}_{wu}^{(t-1)}),$$

where $[\mathbf{x}_u; \mathbf{s}_u]$ is a concatenation of $\mathbf{x}_u$ and $\mathbf{s}_u$ so as to represent comprehensive node information, and W_i^n's (i=1,2, 3,4) are learnable parameter matrices. Similarly to the messages in GMPN, $\mathbf{m}_{uv}^{(t)}$ encodes the information of all length-t paths passing through edge e_{uv} to n_v in the tree. After t_n iterations, the node embedding $\mathbf{n}_v$ is updated as follows:

$$\mathbf{n}_v = \mathrm{ReLU}(U_1^n \mathrm{ReLU}(U_2^n [\mathbf{x}_v; \mathbf{s}_v]) + U_3^n \sum_{n_u \in \mathcal{N}(n_v)} \mathbf{m}_{uv}^{(1\cdots t_n)}), \quad (3)$$

where U_i^n's (i=1,2,3) are the learnable parameter matrices.

Step 3. Difference Embedding (DE)

The difference embedding between M_x and M_y is calculated by pooling the node embeddings from $\mathcal{T}_x$ and $\mathcal{T}_y$ as follows:

$$\mathbf{h}_{xy}^{-} = \sum_{n_x \in \{\mathcal{V}_x \setminus \mathcal{V}_y\} \cup \{n_d \in \mathcal{V}_x\}} \mathbf{n}_x, \qquad \mathbf{h}_{xy}^{+} = \sum_{n_y \in \{\mathcal{V}_y \setminus \mathcal{V}_x\} \cup \{n_d \in \mathcal{V}_y\}} \mathbf{n}_y,$$

where $\mathbf{n}_x$'s/$\mathbf{n}_y$'s are the embeddings of nodes only appearing in and learned from $\mathcal{T}_x/\mathcal{T}_y$ via TMPN. Note that n_d in the above equations is the site of disconnection, and both $\mathcal{T}_x$ and $\mathcal{T}_y$ have the common node n_d. Thus, $\mathbf{h}_{xy}^-$ essentially represents the fragment that should be removed from M_x at n_d and $\mathbf{h}_{xy}^+$ represents the fragment that should be attached to M_x at n_d afterwards in order to modify M_x into M_y. We will discuss how to identify n_d, and the removed and new attached fragments at n_d in M_x and M_y later in Section "Molecular Difference Decoder (Modof-decoder)".

As in VAE,[49] we map the two difference embeddings $\mathbf{h}_{xy}^-$ and $\mathbf{h}_{xy}^+$ into two normal distributions by computing the mean and log variance with fully connected layers $\mu(\cdot)$ and $\Sigma(\cdot)$. We then sample the latent vectors $\mathbf{z}_{xy}^-$ and $\mathbf{z}_{xy}^+$ from these two distributions and concatenate them into one latent vector $\mathbf{z}_{xy}$, that is,

$$\mathbf{z}_{xy}^- \sim N(\mu^-(\mathbf{h}_{xy}^-), \Sigma^-(\mathbf{h}_{xy}^-)), \qquad \mathbf{z}_{xy}^+ \sim N(\mu^+(\mathbf{h}_{xy}^+), \Sigma^+(\mathbf{h}_{xy}^+)),$$
$$\mathbf{z}_{xy} = [\mathbf{z}_{xy}^-; \mathbf{z}_{xy}^+]. \tag{4}$$

Thus, $\mathbf{z}_{xy}$ encodes the difference between M_x and M_y.

Molecular Difference Decoder (Modof-decoder)

Following the autoencoder idea, Modof decodes the difference embedding $\mathbf{z}_{xy}$ (Eqn 4) into edit operations that change M_x into M_y. Specifically, Modof first predicts a node n_d in $\mathcal{T}_x$ as the disconnection site. This node will split $\mathcal{T}_x$ into several fragments, and the number of the resulted fragments depends on the number of n_d's neighboring nodes $\mathcal{N}(n_d)$. Modof then predicts which fragments to remove from M_x, and merges the remaining fragments with n_d into an intermediate representation $M^* = (\mathcal{G}^*, \mathcal{T}^*)$. After that, Modof attaches new fragments sequentially starting from n_d to $(\mathcal{G}^*, \mathcal{T}^*)$. The decoding process (Algorithm S2 in Section S14) has the following 4 steps.

Step 1. Disconnection Site Prediction (DSP)

Modof predicts a disconnection score for each $\mathcal{T}_x$'s node n_u as follows,

$$f_d(n_u) = (\mathbf{w}^d)^\mathsf{T} \tanh\left(W_1^d \mathbf{n}_u + W_2^d \mathbf{z}\right), \forall n_u \in \mathcal{V}_x, \tag{5}$$

where $\mathbf{n}_u$ is n_u's embedding (Eqn 3) in $\mathcal{T}_x$, $\mathbf{w}^d$ and W_i^d's (i=1,2) are learnable parameter vector and matrices, respectively. The node with the largest disconnection score is predicted as the disconnection site n_d. Intuitively, Modof considers the neighboring or local structures of n_u (in $\mathbf{n}_u$) and "how likely" edit operations (represented by $\mathbf{z}$) can be applied at n_u. To learn f_d, Modof uses the negative log likelihood of ground-truth disconnection site in tree $\mathcal{T}_x$ as the loss function.

Step 2. Removal Fragment Prediction (RFP)

Next, Modof predicts which fragments separated by n_d should be removed from $\mathcal{T}_x$. For each node n_u connected to n_d, Modof predicts a removal score as follows,

$$f_r(n_u) = \sigma((\mathbf{w}^r)^\mathsf{T} \mathrm{ReLU}(W_1^r \mathbf{n}_u + W_2^r \mathbf{z}^-)), \forall e_{ud} \in \mathcal{E}_x, \tag{6}$$

where $\sigma(\cdot)$ is sigmoid function, $\mathbf{w}^r$ and W_i^r's (i=1,2) are learnable parameter vector and matrices, respectively. The fragment with a removal score greater than 0.5 is predicted to be removed. Thus, there could be multiple or no fragments removed. Intuitively, Modof considers the local structures of the fragment (i.e., $\mathbf{n}_u$) and "how likely" this fragment should be removed (represented by $\mathbf{z}^-$). To learn f_r, Modof minimizes binary cross entropy loss to maximize the predicted scores of ground-truth removed fragments in $\mathcal{T}_x$.

Step 3. Intermediate Representation (IMR)

After fragment removal, Modof merges the remaining fragments together with the disconnection site n_d into an intermediate representation $M^* = (\mathcal{G}^*, \mathcal{T}^*)$. M^* may not be a valid molecule after some fragments are removed (some bonds are broken). It represents the scaffold of M_x that should remain unchanged during the optimization. Modof first removes a fragment in order to identify such a scaffold and then adds a fragment to the scaffold to modify the molecule.

Step 4. New Fragment Attachment (NFA)

Modof modifies M^* into the optimized M_y by attaching a new fragment (Algorithm S3 in Section S14). Modof uses the following four predictors to sequentially attach new nodes to $\mathcal{T}^*$. The predictors will be applied iteratively, starting from n_d, on each newly attached node in $\mathcal{T}^*$. The attached new node in the t-th step is denoted as $n^{*(t)}$ ($n^{*(0)} = n_d$), and the corresponding molecular graph and tree are denoted as $\mathcal{G}^{*(t)}$ ($\mathcal{G}^{*(0)} = \mathcal{G}^*$) and $\mathcal{T}^{*(t)}$ ($\mathcal{T}^{*(0)} = \mathcal{T}^*$), respectively.

Step 4.1. Child Connection Prediction (NFA-cp) Modof first predicts whether $n^{*(t)}$ should have a new child node attached to it, with the probability calculated as follows:

$$f_c(n^{*(t)}) = \sigma((\mathbf{w}^c)^\mathsf{T} \mathrm{ReLU}(W_1^c \mathbf{n}^{*(t)} + W_2^c \mathbf{z}^+)), \tag{7}$$

where $\mathbf{n}^{*(t)}$ is the embedding of node $n^{*(t)}$ learned over $(\mathcal{T}^{*(t)}, \mathcal{G}^{*(t)})$ (Eqn 3), $\mathbf{z}^+$ (Eqn 4) indicates "how much" $\mathcal{T}^{*(t)}$ should be expanded, and $\mathbf{w}^c$ and W_i^c's (i=1,2) are learnable parameter vector and matrices. If $f_c(n^{*(t)})$ is above 0.5, Modof predicts that $n^{*(t)}$ should have a new child node and thus child node type prediction will follow; otherwise, the optimization process stops at $n^{*(t)}$. To learn f_c, Modof minimizes a binary cross entropy loss to maximize the probabilities of ground-truth child nodes. Note that $n^{*(t)}$ may have multiple children, and therefore, once a child is generated as in the following steps and attached to $\mathcal{T}^{*(t)}$, another child connection prediction will be conducted at $n^{*(t)}$ with the updated

embedding $\mathbf{n}^{*(t)}$ over the expanded $(\mathcal{T}^{*(t)}, \mathcal{G}^{*(t)})$. The above process will be iterated until $n^{*(t)}$ is predicted to have no more children.

Step 4.2. Child Node Type Prediction (NFA-ntp) The new child node of $n^{*(t)}$ is denoted as n_c. Modof predicts the type of n_c by calculating the probabilities of all types of the nodes that can be attached to $n^{*(t)}$ as follows:

$$f_l(n_c) = \text{softmax}(U^l \times \text{ReLU}(W_1^l \mathbf{n}^{*(t)} + W_2^l \mathbf{z}^+)), \tag{8}$$

where softmax$(\cdot)$ converts a vector of values into probabilities, U^l and W_i^l's (i=1,2) are learnable matrices. Modof assigns the new child n_c the node type $\mathbf{x}_c$ corresponding to the highest probability. Modof learns f_l by minimizing cross entropy to maximize the likelihood of true child node types.

Step 4.3. Attachment Point Prediction (NFA-app) If node $n^{*(t)}$ is predicted to have a child node n_c, the next step is to connect $n^{*(t)}$ and n_c. If $n^{*(t)}$ and n_c share one or multiple atoms (e.g., $n^{*(t)}$ and n_c form a fused ring and thus share two adjacent atoms) that can be unambiguously determined as the attachment point(s) based on chemical rules, Modof will connect $n^{*(t)}$ and n_c via the atom(s). Otherwise, if $n^{*(t)}$ and n_c have multiple connection configurations, Modof predicts the attachment atoms at $n^{*(t)}$ and n_c, respectively.

Step 4.3.1. Attachment Point Prediction at Parent Node (NFA-app-p) Modof scores each candidate attachment point at parent node $n^{*(t)}$, denoted as a_p^*, as follows,

$$g_p(a_p^*) = (\mathbf{w}^p)^\mathsf{T} \tanh\left(W_1^p \mathbf{a}_p^* + W_2^p \mathbf{x}_c + W_3^p \times \text{ReLU}(U_2^n[\mathbf{x}^{*(t)}; \tilde{\mathbf{s}}^{*(t)}]) + W_4^p \mathbf{z}^+\right), \tag{9}$$

where $\mathbf{a}_p^* = \sum_{a_i \in a_p^*} \tilde{\mathbf{a}}_i$ represents the embedding of a_p^* (a_p^* could be an atom or a bond), $\tilde{\mathbf{a}}_i$ is calculated by GMPN over $\mathcal{G}^{*(t)}$; U_2^n is as in Eqn 3; $\tilde{\mathbf{s}}^{*(t)}$ is the sum of the embeddings of all atoms in $n^{*(t)}$ (Eqn 1); and $\mathbf{w}^p$ and W_i^p (i=1,2,3,4) are learnable vector and matrices. Modof intuitively measures "how likely" a_p^* can be attached to n_c by looking at a_p^* its own (i.e., $\mathbf{a}_p^*$), its context in $n^{*(t)}$ (i.e., $\mathbf{x}^{*(t)}$ and neighbors $\tilde{\mathbf{s}}^{*(t)}$), its connecting node n_c (i.e., $\mathbf{x}_c$) and "how much" $n^{*(t)}$ should be expanded (represented by $\mathbf{z}^+$). The candidate with the highest score is selected as the attachment point in $n^{*(t)}$. Modof learns g_p by minimizing the negative log likelihood of ground-truth attachment points.

Step 4.3.2. Attachment Point Prediction at Child Node (NFA-app-c) Modof scores each candidate attachment point at the child node n_c, denoted as a_c^*, as follows:

$$g_c(a_c^*) = (\mathbf{w}^o)^\mathsf{T} \tanh\left(W_1^o \mathbf{a}_c^* + W_2^o \mathbf{x}_c + W_3^o \mathbf{a}_p^* + W_4^o \mathbf{z}^+\right), \tag{10}$$

where $\mathbf{a}_c^* = \sum_{a_i \in a_c^*} \tilde{\mathbf{a}}_i$ represents the embedding of a_c^* (a_c^* could be an atom or a bond) and $\tilde{\mathbf{a}}_i$ is a_i's embedding calculated over n_c via GMPN; $\mathbf{w}^o$ and W_i^o's (i=1,2,3,4) are learnable parameters. Modof intuitively measures "how likely" candidate a_c^* can be attached to a_p^* at $n^{*(t)}$ by looking at a_c^* its own (i.e., $\mathbf{a}_c^*$), the features of a_p^* (i.e., $\mathbf{a}_p^*$), its context in n_c (i.e., $\mathbf{x}_c$) and "how much" $n^{*(t)}$ should be expanded (i.e., $\mathbf{z}^+$). The candidate with the highest score is selected as the attachment point in n_c. Modof learns g_c by minimizing the negative log likelihood of ground-truth attachment points.

Valence Checking In NFA-app, Modof incorporates valence check to only generate and predict legitimate candidate attachment points that do not violate valence laws.

Molecule size constraint Following You *et al.*[9], for plogP optimization, we limit the size of optimized molecules to at most 38 (38 is the maximum number of atoms in the molecules in the ZINC dataset[23]). With this molecule size constraint, Modof can avoid increasing plogP by trivially increasing molecule size, which may have the efforts of improving plogP[50].

Sampling Schemes

In the decoding process, for each M_x, Modof samples twenty times from the latent space of $\mathbf{z}$ and optimize M_x accordingly. Among all decoded molecules satisfying the similarity constraint with M_x, Modof selects the one of best property as its output.

Modof Pipelines

A pipeline of Modof models, denoted as Modof-pipe (Algorithm S4 in Section S14), is constructed with a series of identical Modof models, with the output molecule from one Modof model as the input to the next. Given an input molecule $M^{(t)}$ to the t-th Modof model ($M^{(0)}$=M), Modof first optimizes $M^{(t)}$ into $M^{(t+1)}$ as the output of this model. $M^{(t+1)}$ is then fed into the $(t+1)$-th model if it satisfies the similarity constraint $\text{sim}(M^{(t+1)}, M) > \delta$ and property constraint $\text{plogP}(M^{(t+1)}) > \text{plogP}(M^{(t)})$. Otherwise, $M^{(t)}$ is output as the final result and Modof-pipe stops. In addition to Modof-pipe, which outputs one optimized molecule for each input molecule, Modof-pipe^m is developed to output multiple optimized molecules for each input molecule. Details about Modof-pipe^m are available in Section S2.

The advantages of this iterative, one-fragment-at-one-time optimization process include that 1) it is easier to control intermediate optimization steps so as to result in optimized molecules of desired similarities and properties; 2) it is easier to optimize multiple fragments in a molecule that are far apart; and 3) it follows a rational molecule design process[11] and thus could enable more insights and inform *in vitro* lead optimization.

Model Training

During model training, we apply teacher forcing to feed the ground truth instead of the prediction results to the sequential decoding process. Following the idea of variational autoencoder, we minimize the following loss function to maximize the likelihood $P(M_y|\mathbf{z}, M_x)$. Thus, the optimization problem is formulated as follows,

$$\min_{\boldsymbol{\Theta}} -\beta D_{\text{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|M_x, M_y) \| p_{\boldsymbol{\theta}}(\mathbf{z})) + E_{q_{\boldsymbol{\phi}}(\mathbf{z}|M_x, M_y)}[\log p_{\boldsymbol{\theta}}(M_y|\mathbf{z}, M_x)], \tag{11}$$

where $\boldsymbol{\Theta}$ is the set of parameters; $q_{\boldsymbol{\phi}}()$ is an estimated posterior probability function (Modof-encoder); $p_{\boldsymbol{\theta}}(M_y|\mathbf{z}, M_x)$ is the probabilistic decoder representing the likelihood of generating M_y given the latent embedding $\mathbf{z}$ and M_x; and the prior $p_{\boldsymbol{\theta}}(\mathbf{z})$ follows $\mathcal{N}(\mathbf{0}, \mathbf{I})$. In the above problem, $D_{\mathrm{KL}}()$ is the KL divergence between $q_{\boldsymbol{\phi}}()$ and $p_{\boldsymbol{\theta}}()$. Specifically, the second term represents the prediction or empirical error, defined as the sum of all the loss functions in the above six predictions (Eqn 5-10). We use AMSGRAD[51] to optimize the learning objective.

Data Availability

The data used in this manuscript is made publicly available at Chen *et al.*[52] and the link https://github.com/ziqi92/Modof.

Code Availability

The code for Modof, Modof-pipe and Modof-pipe^m is made publicly available at Chen *et al.*[52] and the link https://github.com/ziqi92/Modof.

Acknowledgements

This project was made possible, in part, by support from the National Science Foundation grant numbers (IIS-1855501, X.N.; IIS-1827472, X.N.; IIS-2133650, X.N., S.P.; OAC-2018627, S.P.), and the National Library of Medicine (grant numbers 1R01LM012605-01A1, X.N.; and 1R21LM013678-01, X.N.), an AWS Machine Learning Research Award (X.N.) and The Ohio State University President's Research Excellence program (X.N.). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of the funding agencies. We thank Dr. Xiaoxue Wang and Dr. Xiaolin Cheng for their constructive comments.

Author Contributions

X.N. conceived the research; X.N. and S.P. obtained funding for the research, and co-supervised Z.C.; Z.C., M.R.M., S.P. and X.N. designed the research; Z.C. and X.N. conducted the research, including data curation, formal analysis, methodology design and implementation, result analysis and visualization; Z.C. drafted the original manuscript; M.R.M. provided comments on the original manuscript; Z.C., X.N. and S.P. conducted the manuscript editing and revision; all authors reviewed the final manuscript.

Competing Interests

M.R.M. was employed by the company NEC Labs America. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Jorgensen, W. L. Efficient drug lead discovery and optimization. *Acc. Chem. Res.* **42**, 724–733 (2009).
2. Verdonk, M. L. & Hartshorn, M. J. Structure-guided fragment screening for lead discovery. *Curr. Opin. Drug Discov. Devel.* **7**, 404 (2004).
3. de Souza Neto, L. R. *et al.* In silico strategies to support fragment-to-lead optimization in drug discovery. *Front. Chem.* **8** (2020).
4. Hoffer, L. *et al.* Integrated strategy for lead optimization based on fragment growing: the diversity-oriented-target-focused-synthesis approach. *J. Med. Chem.* **61**, 5719–5732 (2018).
5. Gerry, C. J. & Schreiber, S. L. Chemical probes and drug leads from advances in synthetic planning and methodology. *Nat. Rev. Drug Discov.* **17**, 333 (2018).
6. Sattarov, B. *et al.* De novo molecular design by combining deep autoencoder recurrent neural networks with generative topographic mapping. *J. Chem. Inf. Model.* **59**, 1182–1196 (2019).
7. Sanchez-Lengeling, B. & Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* **361**, 360–365 (2018).
8. Jin, W., Barzilay, R. & Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. vol. 80 of *Proceedings of Machine Learning Research*, 2323–2332 (Stockholmsmässan, Stockholm Sweden, 2018).
9. You, J., Liu, B., Ying, Z., Pande, V. & Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems 31*, 6410–6421 (2018).
10. Murray, C. & Rees, D. The rise of fragment-based drug discovery. *Nat. Chem.* **1**, 187–92 (2009).
11. Hajduk, P. J. & Greer, J. A decade of fragment-based drug design: strategic advances and lessons learned. *Nat. Rev. Drug Discov.* **6**, 211–219 (2007).
12. Shi, C. *et al.* Graphaf: a flow-based autoregressive model for molecular graph generation. In *8th International Conference on Learning Representations, Addis Ababa, Ethiopia, April 26-30, 2020* (2020).
13. Zang, C. & Wang, F. Moflow: An invertible flow model for generating molecular graphs. In Gupta, R., Liu, Y., Tang, J. & Prakash, B. A. (eds.) *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, 617–626 (2020).
14. Jin, W., Yang, K., Barzilay, R. & Jaakkola, T. S. Learning multimodal graph-to-graph translation for molecule optimization. In *7th International Conference on Learning Representations, New Orleans, LA, USA, May 6-9, 2019* (2019).
15. Jin, W., Barzilay, R. & Jaakkola, T. S. Hierarchical generation of molecular graphs using structural motifs. In *Proceedings of the 37th International Conference on Machine Learning, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research*, 4839–4848 (2020).
16. Podda, M., Bacciu, D. & Micheli, A. A deep generative model for fragment-based molecule generation. In Chiappa, S. & Calandra, R. (eds.) *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, vol. 108 of *Proceedings of Machine Learning Research*, 2240–2250 (2020).
17. Ji, C., Zheng, Y., Wang, R., Cai, Y. & Wu, H. Graph polish: A novel graph generation paradigm for molecular optimization. *CoRR* **abs/2008.06246** (2020). 2008.06246.
18. Lim, J., Hwang, S.-Y., Moon, S., Kim, S. & Kim, W. Y. Scaffold-based molecular design with a graph generative model. *Chem. Sci.* **11**, 1153–1164 (2020).
19. Ahn, S., Kim, J., Lee, H. & Shin, J. Guiding deep molecular optimization with genetic exploration. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H. (eds.) *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, December 6-12, 2020, virtual* (2020).
20. Nigam, A., Friederich, P., Krenn, M. & Aspuru-Guzik, A. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *8th International Conference on Learning Representations, Addis Ababa, Ethiopia, April 26-30, 2020* (2020).
21. Wildman, S. A. & Crippen, G. M. Prediction of physicochemical parameters by atomic contributions. *J. Chem. Inf. Comput. Sci.* **39**, 868–873 (1999).
22. Ertl, P. & Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminf.* **1**, 8 (2009).
23. Sterling, T. & Irwin, J. J. Zinc 15–ligand discovery for everyone. *J. Chem. Inf. Model.* **55**, 2324–2337 (2015).
24. Gómez-Bombarelli, R. *et al.* Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* **4**, 268–276 (2018).
25. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y. & Martineau, P. An exact graph edit distance algorithm for solving pattern recognition problems. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods - Volume 1*, 271–278 (Setubal, PRT, 2015).
26. Sanfeliu, A. & Fu, K. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Syst. Man Cybern.* **SMC-13**, 353–362 (1983).
27. Lipinski, C. A. Lead-and drug-like compounds: the rule-of-five revolution. *Drug Discov. Today Technol.* **1**, 337–341 (2004).
28. Ghose, A. K., Viswanadhan, V. N. & Wendoloski, J. J. A knowledge-based approach in designing combinatorial or medicinal chemistry libraries for drug discovery. 1. a qualitative and quantitative characterization of known drug databases. *J. Comb. Chem.* **1**, 55–68 (1999).
29. Whiteson, S., Tanner, B., Taylor, M. E. & Stone, P. Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 120–127 (2011).
30. Zhang, C., Vinyals, O., Munos, R. & Bengio, S. A study on overfitting in deep reinforcement learning. *CoRR* **abs/1804.06893** (2018). 1804.06893.
31. Lipinski, C. A., Lombardo, F., Dominy, B. W. & Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Deliv. Rev.* **46**, 3–26 (2001).
32. Rokitskaya, T. I., Luzhkov, V. B., Korshunova, G. A., Tashlitsky, V. N. & Antonenko, Y. N. Effect of methyl and halogen substituents on the transmembrane movement of lipophilic ions. *Phys. Chem. Chem. Phys.* **21**, 23355–23363 (2019).
33. Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S. & Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nat. Chem.* **4**, 90–98 (2012).
34. Olivecrona, M., Blaschke, T., Engkvist, O. & Chen, H. Molecular de-novo design through deep reinforcement learning. *J. Cheminf.* **9** (2017).
35. Kusner, M. J., Paige, B. & Hernández-Lobato, J. M. Grammar variational autoencoder. In Precup, D. & Teh, Y. W. (eds.) *Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6-11 August 2017*, vol. 70 of *Proceedings of Machine Learning Research*, 1945–1954 (2017).
36. De Cao, N. & Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models* (2018).
37. Zhou, Z., Kearnes, S., Li, L., Zare, R. N. & Riley, P. Optimization of molecules via deep reinforcement learning. *Sci. Rep.* **9**, 1–10 (2019).
38. Wainberg, M., Merico, D., Delong, A. & Frey, B. J. Deep learning in biomedicine. *Nat. Biotechnol.* **36**, 829–838 (2018).

39. Kim, S. *et al.* PubChem in 2021: new data content and improved web interfaces. *Nucleic Acids Res.* **49**, D1388–D1395 (2020).
40. Gao, W. & Coley, C. W. The synthesizability of molecules proposed by generative models. *J. Chem. Inf. Model.* (2020).
41. Segler, M. H. S., Preuss, M. & Waller, M. P. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature* **555**, 604–610 (2018).
42. Kishimoto, A., Buesser, B., Chen, B. & Botea, A. Depth-first proof-number search with heuristic edge cost and application to chemical synthesis planning. In Wallach, H. M. *et al.* (eds.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, December 8-14, 2019, Vancouver, BC, Canada*, 7224–7234 (2019).
43. Stokes, J. M. *et al.* A deep learning approach to antibiotic discovery. *Cell* **180**, 688–702.e13 (2020).
44. Liu, J. & Ning, X. Multi-assay-based compound prioritization via assistance utilization: A machine learning framework. *J. Chem. Inf. Model.* **57**, 484–498 (2017).
45. Liu, J. & Ning, X. Differential compound prioritization via bidirectional selectivity push with power. *J. Chem. Inf. Model.* **57**, 2958–2975 (2017).
46. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 1263–1272 (2017).
47. Xu, K., Hu, W., Leskovec, J. & Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations, New Orleans, LA, USA, May 6-9, 2019* (2019).
48. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. In Bengio, Y. & LeCun, Y. (eds.) *2nd International Conference on Learning Representations, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (2014).
49. Wildman, S. A. & Crippen, G. M. Prediction of physicochemical parameters by atomic contributions. *J. Chem. Inf. Comput. Sci.* **39**, 868–873 (1999).
50. Reddi, S. J., Kale, S. & Kumar, S. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018).
51. Chen, Z. A deep generative model for molecule optimization via one fragment modification. http://doi.org/10.5281/zenodo.4667928 (2021).

A Deep Generative Model for Molecule Optimization via One Fragment Modification (Supplementary Information)

S1 Additional Related Work

There exists some limited work that follow the fragment-based drug design as Modof does. Podda *et al.*[16] decomposed molecules into fragments using an algorithm that breaks bonds according to chemical reactions. They then represented each fragment using its SMILES string and generated molecules via a VAE-based model, which sequentially decodes the SMILES strings of the fragments and reassembles the decoded SMILES strings into complete molecules.

Similar to Modof, a recent work named Teacher and Student polish (T&S polish)[17] also proposed to retain the scaffolds and edit the molecules by removing and adding fragments. However, Modof is fundamentally different from T&S polish. T&S polish employs a teacher component to identify the logic rules from training molecules that can transform one molecule to another with better properties. Thus, the logic rules describe an one-to-one mapping between the two molecules. T&S polish then learns from these logic rules in the student component, and uses the student component to polish or modify new molecules. The limitation of T&S polish is that it generates only one modified molecule for each input molecule. However, there could be multiple ways to optimize one molecule, and as suggested in Jin *et al.*,[14] generative models should be able to generate such multiple, diverse optimized molecules for each input molecule. In contrast to T&S polish, Modof samples from a latent 'difference' distribution during testing and thus is able to generate multiple diverse optimized molecules.

In addition to T&S polish, Lim *et al.*[18] also developed a scaffold-based method to generate the molecules from scaffolds. Their method takes a scaffold as input and completes the input scaffold into a molecule through sequentially adding atoms and bonds via a VAE-based model. The limitation of their method is that the retained scaffolds must be cyclic skeletons extracted from training data by removing side-chain atoms. Due to this pre-defined scaffold vocabulary, their model is only able to add side-chain atoms to input scaffolds, and their generated molecules are limited within the chemical subspace shattered by their scaffold vocabulary. In contrast to their method, Modof is able to learn to determine the scaffolds that need to be retained from input test molecules, and completes the identified scaffolds with fragments that could be more complicated than just side-chain atoms. Hence, Modof has the potential to explore more widely in the chemical space for molecule optimization. We could not compare Modof with T&S polish as the T&S polish authors haven't published their code. They also applied a different experimental setting (e.g., sample only once for all the baseline methods and thus lead to underestimation of the baselines) so that we could not directly use their reported results. Issues related to their parameter setting are discussed in Supplementary Information Section S6.

In addition to deep generative models, some genetic algorithm-based methods are also developed to find molecules with better properties. Ahn *et al.*[19] developed a genetic expert-guided learning method (GEGL), in which they used an expert policy to modify molecules through mutation and crossover, and learned a parameterized apprentice policy from good molecules modified by the expert policy for imitation learning. Nigam *et al.*[20] used a genetic algorithm to modify molecules with random mutations defined in Krenn *et al.*,[53] and employed a discriminator to prevent the genetic algorithm from searching the same or similar molecules repeatedly.

S2 Modof-pipe^m Protocol

We further allow Modof-pipe to output multiple, diverse optimized molecules (the corresponding pipeline denoted as Modof-pipe^m) following the below protocol:

1) At the t-th iteration in Modof-pipe^m, Modof optimizes each input molecule into 20 output molecules via 20 times of sampling and decoding.
2) Among all the output molecules from iteration t that satisfy the similarity constraint, the top-5 molecules with the best properties are fed into the next, $(t+1)$-th iteration. The remaining output molecules may still have better properties compared to the input molecule to Modof-pipe^m, but they will not be further optimized by Modof-pipe^m. Note that the top-5 molecules may not always have improved properties (e.g., when all the output molecules do not have improved properties), but they will still be further optimized in the downstream iterations.
3) The above two steps are conducted at each iteration up to five iterations, or until the iteration does not output any molecules (e.g., molecules cannot be decoded, similarity constraints are not satisfied), and then Modof-pipe^m stops.
4) Once Modof-pipe^m has stopped, all the unique molecules output at each iteration that are not further optimized (either not fed into the next iteration, or output at the last iteration) are collected, and the top-20 molecules among them with the best properties will be the output, optimized molecules of Modof-pipe^m.

Algorithm S5 presents the Modof-pipe^m algorithm.

S3 Data Used in plogP Optimization Experiments

Table S1 presents the statistics of the data used in our experiments for plogP optimization.

S3.1 Training Data Representativeness

The training data used in our experiments and the baseline methods are extracted from the widely used ZINC dataset. The ZINC dataset is relatively standard for problems including chemical property prediction[54], chemical synthesis[55],

Table S1 | Data Statistics for plogP Optimization

description	value
#training molecules	104,708
#training (M_x, M_y) pairs	55,686
#validation molecules	200
#test molecules	800
average similarity of training (M_x, M_y) pairs	0.6654
average pairwise similarity between training and test molecules	0.1070
average training molecule size	25.04
average training $\{M_x\}$ size	22.75
average training $\{M_y\}$ size	27.07
average test molecule size	20.50
average $\{M_x\}$ plogP	-0.7362
average $\{M_y\}$ plogP	1.1638
average test molecule plogP	-2.7468
average plogP improvement in training (M_x, M_y) pairs	1.9000

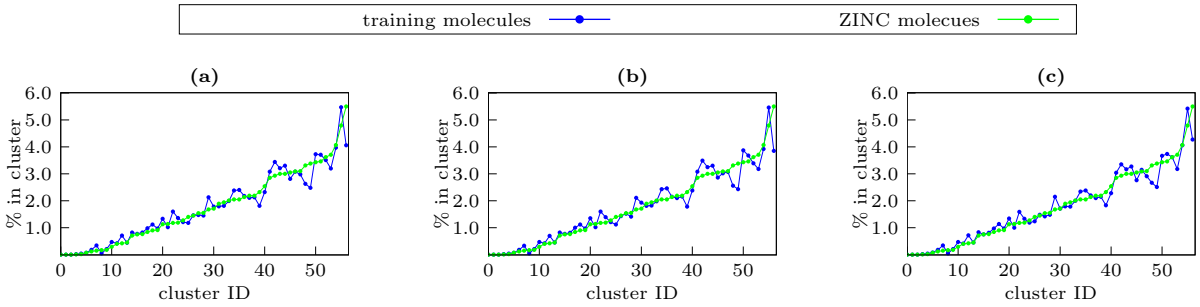


Fig. S1 | Molecule Representativeness for ZINC Chemical Space. a, Training data representativeness. **b,** $\{M_x\}_{\text{trn}}$ data representativeness. **c,** $\{M_y\}_{\text{trn}}$ data representativeness

optimization[14,15], etc. Instead of using the entire ZINC or a random subset of ZINC, in Modof, we used pairs of molecules in ZINC that satisfy a particular structural constraint: the molecules in a pair are only different in structures at one disconnection site. Whether Modof's training data well represent the ZINC chemical space will affect whether Modof can be well generalized in the entire ZINC space.

To analyze the representativeness of Modof's training data, we conducted the following analysis: We clustered the following three groups of molecules all together: (1) the molecules in Modof's training pairs that have bad properties, denoted as $\{M_x\}_{\text{trn}}$; (2) the molecules in Modof's training pairs that have good properties, denoted as $\{M_y\}_{\text{trn}}$; and (3) the rest, all the ZINC molecules that are not in $\{M_x\}_{\text{trn}}$ and $\{M_y\}_{\text{trn}}$, denoted as $\{M\}_{\text{ZINC}}$. In total, we had 324,949 molecules to cluster. We first represented each molecule using its canonical SMILE string, and generated a 2,048-dimension binary Morgan fingerprint based on the SMILE string. We then clustered the 324,949 molecules using the CLUTO[56] clustering software. CLUTO constructs a graph among the molecules, in which each molecule is connected to its nearest neighbors defined by molecule similarities calculated via Tanimoto coefficient over molecule fingerprints. Please note that this is exactly the same molecule similarity calculation used in Modof.

Fig. S1 presents the results from 56 clusters (50 clusters and 6 disconnected components in the nearest-neighbor graphs). In Fig. S1, the clusters are sorted based on their size, and the y axis represents the percentage of molecules in $\{M\}_{\text{ZINC}}$, training data (i.e., $\{M_x\}_{\text{trn}} \cup \{M_y\}_{\text{trn}}$), $\{M_x\}_{\text{trn}}$ or $\{M_y\}_{\text{trn}}$ that fall within in each cluster. Fig. S1 shows that Modof's training data, $\{M_x\}_{\text{trn}}$ and $\{M_y\}_{\text{trn}}$ have data distributions similar to ZINC data in each of the clusters. We also conducted six paired t-tests over the data distributions among all the group pairs between $\{M\}_{\text{ZINC}}$, $\{M_x\}_{\text{trn}} \cup \{M_y\}_{\text{trn}}$, $\{M_x\}_{\text{trn}}$ and $\{M_y\}_{\text{trn}}$. The t-tests show no statistically significant difference in data distributions over clusters among the four groups of molecules, with all the p-values close to 1.0. This indicates that Modof's training data actually well represent the entire ZINC data, and thus Modof is generalizable and applicable to ZINC molecules outside Modof's training data. We also tried different numbers of clusters (e.g., 100, 200), and the above conclusions remain the same.

S4 Graph Edit Path Identification for Training Data Generation

Graph edit distance between tree $\mathcal{T}_x$ and $\mathcal{T}_y$ is defined as the minimum cost to modify $\mathcal{T}_x$ into $\mathcal{T}_y$ with the following graph edit operations:

- Node addition: add a new labeled node into $\mathcal{T}_x$;
- Node deletion: delete an existing node from $\mathcal{T}_x$;
- Edge addition: add a new edge between a pair of nodes in $\mathcal{T}_x$; and
- Edge deletion: delete an existing edge between a pair of nodes in $\mathcal{T}_x$.

Particularly, we did not allow node or edge substitutions as they can be implemented via deletion and addition operations. We identified the optimal graph edit paths using the DF-GED algorithm[25] provided by a widely-used package NetworkX.[57] To identify disconnection sites, we denoted the common nodes between $\mathcal{V}_x$ and $\mathcal{V}_y$ as matched nodes $\mathcal{M}$ (i.e., $\mathcal{M} = \mathcal{V}_x \cap \mathcal{V}_y$), nodes only in $\mathcal{V}_x$ as the removal nodes $\mathcal{D}$ (i.e., $\mathcal{D} = \mathcal{V}_x \setminus \mathcal{V}_y$), and the nodes only in $\mathcal{V}_y$ as the new nodes $\mathcal{J}$ (i.e., $\mathcal{J} = \mathcal{V}_y \setminus \mathcal{V}_x$), all with respect to $\mathcal{T}_x$. Therefore, the disconnection sites will be the matched nodes in $\mathcal{T}_x$ that are also connected with a new node or a removal node, that is, $\{n_d | (n_d \in \mathcal{M}) \wedge (\mathcal{N}(n_d) \cap (\mathcal{D} \cup \mathcal{J}) \neq \emptyset)\}$.

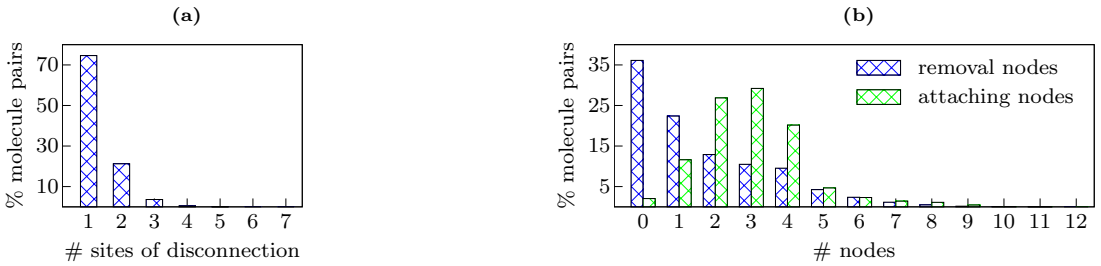


Fig. S2 | Distributions of Modification Operations. a, Distribution of disconnection site counts. **b,** Distribution of removal and attaching node counts.

S5 Disconnection Site Analysis in plogP Training Data

The key idea of Modof is to modify one fragment at a time under the similarity and property constraints. This is based on the assumption that for a pair of molecules with a high similarity (e.g., $\text{sim}(M_x, M_y) > 0.6$), they are very likely to have only one (small) fragment different, and thus one site of disconnection. Fig. S2a presents the number of disconnection sites among the 75K pairs of molecules in the benchmark dataset provided by Jin *et al.*,[15] in which each pair has similarity above 0.6. The distribution in Fig. S2a shows that when the molecule similarity is high (e.g., 0.6 in the benchmark dataset), most of the molecule pairs (i.e., 74.4%) have only one disconnection site, some of the pairs have two (i.e., 21.1%), and only a few have three. This indicates that one-fragment modification at a time is a rational idea and directly applicable to the majority of the optimization cases. Even though there could be more disconnection sites, Modof-pipe and Modof-pipe^m allow multiple-fragment optimization via multiple one-fragment optimizations.

Fig. S2b presents the number of nodes (as in junction tree representations) that need to be removed from the disconnection sites at M_x ($\mathcal{T}_x$), and the number of nodes that need to be attached at the sites afterwards, in Jin's benchmark dataset.[15] The figures show that on average, more nodes will be attached to the disconnection sites compared to the removal nodes. This indicates that the optimized molecules with better plogP will become larger, as we have observed in our and other's methods.

S6 Discussion on Parameter Settings in plogP Optimization

Among all the methods that involve random sampling, JTNN, HierG2G and our method Modof sample 20 times for each test molecule and thus produce 20 optimized candidates, and identify the best one among these 20 candidates. However, GraphAF optimized each test molecule for 200 times and reported the best among those 200 candidates. Thus, it is unclear if the overall performance of GraphAF is largely due to the many times of optimization (and thus a larger pool of optimized candidates) or the model that does learn how to optimize. T&S polish[17] forced all their compared baseline models including JTNN and GCPN to sample only once for each test molecule, because T&S polish can only modify one input molecule into one output. This might not be appropriate either since it artificially underestimated the performance of the baseline models. Instead, it would be fair to compare the output from T&S polish with the best output from the baseline methods.

S7 Molecule Similarity Calculation

All baselines except JT-VAE in Table 1 in the main manuscript use binary Morgan fingerprints with radius of 2 and 2,048 bits to represent the presence or absence of particular, pre-defined substructures in molecules. Using such binary fingerprints in molecule similarity calculation may overestimate molecule similarities. An example is presented in Fig. S3, where the Tanimoto similarity from binary Morgan fingerprints (sim_b) of the two molecules is 0.644, but they look sufficiently different, with the similarity from Morgan fingerprints of substructure counts (sim_c) only 0.393. According to our experimental results (e.g., Fig. 1c and Fig. 1d in the main manuscript) and fragment analysis later in Section S10, we observed that aromatic rings could contribute to large plogP values and thus would be attached to the optimized molecules. Using binary Morgan fingerprints in calculating molecule similarities in this case would easily lead to the solution that many aromatic rings will be attached for molecule optimization, while still satisfying the similarity constraint due to the similarity overestimation, but result in very large molecules that are less drug-like.[28] Therefore, to prevent the model from generating such molecules, we could either consider Morgan fingerprints of substructure counts, or limit the size of optimized molecules. In Modof, we used the same binary Morgan fingerprints as in the baselines and added an additional constraint to limit the size of optimized molecules to be at most 38 (38 is the maximum number of atoms in the molecules in the ZINC dataset).

S8 Comparison Fairness among Existing Methods

S8.1 Discussion on plogP Test Sets

Note that in Table 1 in the main manuscript, the results of GraphAF and MoFlow are lower than those reported in the respective paper. This is because in their papers, they used a different test set rather than the benchmark test set, and their test molecules were much easier to optimize, which would lead to an unfair comparison with other baseline methods. In our experiments, we tested GraphAF and MoFlow on the benchmark test set and reported the results in Table 1 in the main manuscript.

The benchmark test set consists of 800 molecules that have the lowest plogP values in ZINC test set (ZINC test set

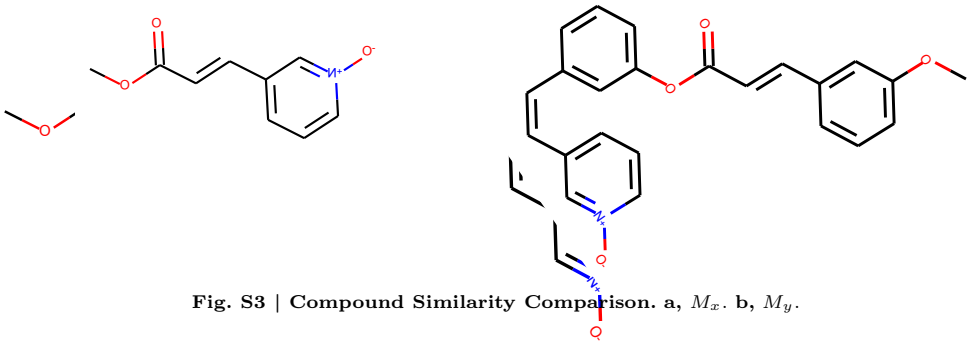


Fig. S3 | Compound Similarity Comparison. a, M_x. **b,** M_y.

Figure A2: Compound Similarities

Figure A2: Compound Similarities

Table S2 | Modof-pipe Performance on Optimizing plogP

δ	t	#in%	#p%	#n%	#z%	property improvement			molecule sim		avgsim w. Trn			top-10 sim w. Trn			sim$_t$
						p$_t$±std	n$_t$±std	p±std	sim$_t$±std	sim±std	all	Trn$_x$	Trn$_y$	all	Trn$_x$	Trn$_y$	M_y
0.0	1	100.00	99.62	0.00	0.38	5.12±1.73	0.00±0.00	5.10±1.76	0.43±0.13	0.44±0.13	0.119	0.112	0.125	0.369	0.333	0.355	0.146
	2	99.62	84.88	0.62	14.12	1.94±1.26	-1.55±1.12	6.74±2.14	0.63±0.19	0.26±0.15	0.116	0.107	0.125	0.365	0.319	0.358	0.173
	3	84.88	55.00	1.75	28.12	1.07±0.98	-0.52±0.59	7.33±2.23	0.76±0.21	0.22±0.15	0.114	0.103	0.124	0.365	0.312	0.360	0.194
	4	55.75	32.12	0.38	23.25	0.62±0.76	-0.60±0.78	7.53±2.28	0.84±0.18	0.22±0.15	0.111	0.099	0.122	0.361	0.306	0.358	0.213
	5	32.50	16.50	0.38	15.62	0.49±0.61	-1.68±1.98	7.61±2.30	0.86±0.17	0.21±0.15	0.107	0.095	0.119	0.356	0.299	0.353	0.225
0.2	1	100.00	99.62	0.00	0.38	4.92±1.58	0.00±0.00	4.91±1.60	0.46±0.13	0.46±0.12	0.117	0.111	0.123	0.364	0.334	0.346	0.138
	2	99.62	68.75	2.38	28.50	1.54±0.97	-0.78±0.84	5.97±1.73	0.71±0.19	0.36±0.11	0.117	0.109	0.125	0.351	0.317	0.340	0.154
	3	70.12	28.62	3.25	38.25	0.75±0.71	-0.67±0.87	6.18±1.76	0.86±0.15	0.34±0.11	0.117	0.108	0.125	0.348	0.312	0.340	0.166
	4	29.88	8.38	1.25	20.25	0.53±0.48	-0.28±0.28	6.22±1.77	0.92±0.13	0.34±0.11	0.115	0.106	0.124	0.345	0.307	0.338	0.175
	5	9.25	1.75	0.25	7.25	0.32±0.22	-0.17±0.14	6.23±1.77	0.96±0.09	0.34±0.11	0.113	0.103	0.122	0.344	0.308	0.338	0.177
0.4	1	100.00	99.12	0.00	0.88	4.50±1.32	0.00±0.00	4.47±1.38	0.53±0.11	0.53±0.11	0.114	0.110	0.118	0.366	0.338	0.344	0.131
	2	99.12	34.25	3.00	61.88	1.36±0.95	-0.65±0.56	4.93±1.49	0.75±0.22	0.49±0.09	0.114	0.110	0.119	0.360	0.331	0.341	0.134
	3	37.62	8.62	0.75	28.25	0.65±0.64	-0.77±0.67	4.99±1.52	0.91±0.14	0.48±0.09	0.115	0.110	0.120	0.360	0.329	0.343	0.146
	4	9.25	1.88	0.12	7.25	0.57±0.54	-0.29±0.00	5.00±1.53	0.93±0.12	0.48±0.09	0.114	0.109	0.119	0.358	0.328	0.341	0.149
	5	2.25	0.38	0.00	1.88	0.37±0.06	0.00±0.00	5.00±1.53	0.97±0.06	0.48±0.09	0.112	0.107	0.117	0.355	0.328	0.335	0.142
0.6	1	100.00	84.25	0.12	15.62	3.01±1.16	-0.31±0.00	2.54±1.53	0.60±0.23	0.66±0.05	0.109	0.107	0.111	0.383	0.360	0.351	0.117
	2	84.25	14.62	1.88	67.75	1.16±0.92	-0.69±0.76	2.71±1.68	0.83±0.17	0.66±0.05	0.113	0.111	0.116	0.382	0.357	0.355	0.128
	3	16.00	1.00	0.25	14.75	0.84±0.67	-0.03±0.00	2.72±1.68	0.94±0.11	0.65±0.05	0.113	0.109	0.117	0.369	0.342	0.346	0.136
	4	1.25	0.12	0.00	1.12	0.38±0.00	0.00±0.00	2.72±1.68	0.92±0.14	0.65±0.05	0.117	0.112	0.121	0.351	0.322	0.334	0.147
	5	0.12	0.00	0.00	0.12	0.00±0.00	0.00±0.00	2.72±1.68	0.00±0.00	0.65±0.05	0.111	0.107	0.115	0.271	0.255	0.262	0.000

Columns represent: "t": the iteration; '#in%": the number of input molecules in each iteration in percentage over all the testing molecules; "#p"/"#n"/"#z%": the percentage of molecules optimized with better/worse/same properties; "p$_t$"/"n$_t$": property improvement/decline in the t-th iteration; "p": the overall property improvement up to the t-th iteration; "sim$_t$"/"sim": the similarities between the molecules before and after optimization in/up to the t-th iteration; "avgsim w. Trn"/"top-10 sim w. Trn": the average similarities with all/top-10 most similar training molecules; "all"/"Trn$_x$"/"Trn$_y$": the comparison molecules identified from all/poor-property/good-property training molecules; "sim$_t$ M_y": the average pairwise similarities among optimized molecules.

was split by Gómez-Bombarelli *et al.*[24]). The plogP values of these molecules are in range [-11.02, -0.56], with an average value -2.75±1.52. The test set used in GraphAF and MoFlow consists of 800 molecules that have the lowest plogP values from the *entire* ZINC dataset, not only from the ZINC *test* set. The plogP values of these molecules are in range [-62.52, 2.42], with an average score -12.00±5.89. That is, the GraphAF and MoFlow's test molecules have much worse properties compared to those in the benchmark test set, and they are much easier to optimize with larger property improvement. Due to the different test sets, the results reported in GraphAF and MoFlow are not comparable to those reported in other baseline methods.

For GraphAF, we fine-tuned their pre-trained model with a set of 10K molecules that do not overlap with the benchmark test set. We tested MoFlow using the trained model provided by its authors (note that MoFlow training learns molecule latent representations, not how to optimize molecules; molecule optimization is conducted during testing via gradient-ascent search in the latent representation space). The results show that unfortunately, GraphAF and MoFlow do not outperform GCPN, JTNN, HierG2G and our method Modof on the benchmark test data.

S8.2 Discussion on Reinforcement Learning Settings

GCPN, MolDQN and GraphAF are reinforcement learning-based methods. They all used the 800 test molecules (benchmark test set for GCPN and MolDQN, and a different non-benchmark test set for GraphAF) to either directly train a model or fine-tune a pre-trained model to optimize the test molecules. Therefore, these models are specific to their test set, and would suffer from overfitting issues and not generalize to new molecules. They may also have the issue of not really learning but essentially memorizing optimization actions/paths. Previous studies have analyzed this so-called environment overfitting problem[29,30] in reinforcement learning. They concluded that using a test set with overlapping samples in the training set (i.e., non-isolated test set) can lead to artificially high performance by simply memorizing the sequences of actions, and suggested that reinforcement learning should use an isolated test set that is completely disjoint with training set. The issues with non-isolated test sets remain true for GCPN, MolDQN and GraphAF.

In our experiments on GraphAF, instead of using the test set to fine-tune the pre-trained GraphAF model, we sampled 10K molecules from ZINC dataset that do not overlap with the benchmark test molecules and also have the property range similar with the test molecules (i.e., [-11.0, -0.5]). We then fine-tuned the pre-trained GraphAF model provided by the authors over these 10K molecules and used the fine-tuned model to optimize each test molecule. The results in Table 1 in the main manuscript demonstrate that unfortunately, GraphAF model does not outperform other baseline methods over the isolated benchmark test molecules.

S9 Additional Experimental Results on plogP Optimization

S9.1 Overall Pipeline Performance

Table S2 presents the Modof-pipe performance in each of its iterations under δ=0.0, 0.2, 0.4 and 0.6, where the output of Modof-pipe at t-th iteration for input molecule M_x is denoted as $M_x^{(t)}$. Without similarity constraints (i.e., δ=0.0), Table S2 shows that 84.88% of the molecules can go through three iterations in Modof-pipe, and fewer molecules go through further iterations. The property improvement (i.e., "p_t±std") gets slower in later iterations as it becomes more difficult to optimize a good molecule. This is also indicated by the increasing molecule similarities in later iterations (i.e., large sim_t±std values over iterations). Still, after each iteration, the overall property improvement out of the pipeline (i.e., "p±std") increases (e.g., 7.61±2.30 after iteration 5), while the overall molecule similarity (i.e., "sim±std") decreases over iterations.

With similarity constraint δ=0.2, Table S2 shows trends very similar as to those with δ=0.0. In addition, at δ=0.2, most molecules can be optimized within first three iterations and do not go through further iterations (e.g., at t=4, "#in%"=29.88% and "#p%"=8.38%). This could be due to that the decoded molecules in later iterations do not satisfy the similarity constraint. Accordingly, the output optimized molecules may not necessarily be of the best properties, and there are a few output molecules with even declined properties ("#n%"). With even higher similarity threshold δ=0.4 or 0.6, Table S2 shows that even fewer molecules (#in%) can be further optimized and property improvement from each iteration is also smaller.

S9.2 Molecule Similarity Comparison

Table S2 also presents the similarities between the optimized molecules and training molecules (i.e., "avgsim w. Trn" and "top-10 sim w. Trn"), and the average pairwise similarities among optimized molecules (i.e., "sim_t M_y"). Table S2 shows that as Modof-pipe optimizes a molecule more, the average pairwise similarities between the optimized and all the training molecules ("avgsim w. Trn") almost remain the same with small values. This could be due to the effect that there are many training modules (104,708) and the all-pairwise similarities are smoothed out. However, the average similarities between the optimized molecules and their top-10 most similar training molecules ("top-10 sim w. Trn") decrease. This indicates that Modof generates new molecules that are in general different from training molecules. In addition, the optimized molecules out of each iteration have low similarities around 0.14-0.18 ("sim_t M_y"), indicating their diversity. The optimized molecules become slightly more similar to one another. This could indicate that the optimized molecules also share certain similarities due to their good plogP properties (e.g., aromatic proportion[58]).

S9.3 Optimized Molecule Size Analysis

Table S3 presents the average size of the optimized molecules in each iteration with δ=0.0, 0.2, 0.4 and 0.6. Without any similarity constraints (i.e., δ=0.0), the average size of the optimized molecules keeps increasing as large as 36.20 after 5

Table S3 | Optimized Molecule Size

iter	δ=0.0		δ=0.2		δ=0.4		δ=0.6	
	$\#a_x$	$\#a_y$	$\#a_x$	$\#a_y$	$\#a_x$	$\#a_y$	$\#a_x$	$\#a_y$
1	20.51	32.70	20.51	32.64	20.51	32.13	20.51	25.25
2	32.82	34.45	32.71	35.09	32.26	33.60	29.08	29.78
3	34.26	35.28	35.16	35.83	34.63	35.02	33.03	33.30
4	34.91	35.83	36.00	36.23	35.05	35.43	36.20	36.30
5	35.51	36.20	36.41	36.50	35.78	35.94	38.00	38.00

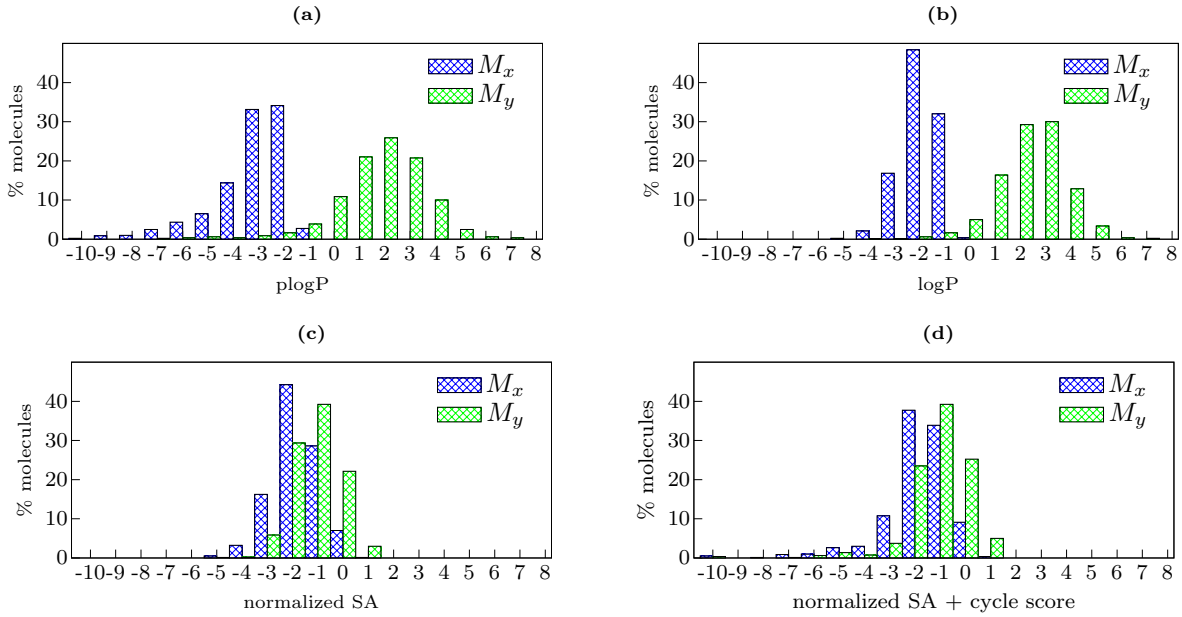


Fig. S4 | Test Molecule Property Distributions before and after Modof-pipe Optimization. **a,** plogP distribution. **b,** logP distribution **c,** Normalized synthetic accessibility distribution. **d,** Synthetic accessibility + cycle score distribution.

iterations of optimization (note that the optimized molecules are always constrained to have fewer than 38 atoms). In addition, the number of added atoms (i.e., $\#a_y - \#a_x$) becomes smaller in later iterations (e.g., for δ=0.0, 12.19, 1.63, 1.02, 0.92, 0.69 from iteration 1 to 5, respectively). This indicates that in later iterations, Modof identifies fewer fragments that should be removed from the input molecules. In the meantime, the constraint of optimized molecule size ensures that the newly added fragments in later iterations should not increase the size of optimized molecules substantially. This also explains the less property improvement ($p_t \pm$std) with higher similarity constraint (e.g., δ=0.6) in Table S2. With similarity constraint δ=0.2, 0.4 and 0.6, the size of optimized molecules exhibits trends similar to those with δ=0.0.

S9.4 Improvement on Different Components of plogP

Recall that plogP has three components: octanol-water partition coefficients (logP), synthetic accessibility (SA; measured by normalized SA scores[22]) and ring size (measured as cycle scores $= -(\min(0, \max(\text{ring size}) - 6))$, that is, plogP = logP + SA score + cycle score. Fig. S4 presents the distributions of plogP values (Fig. S4a), logP values (Fig. S4b), SA scores (Fig. S4c) and the combined values of SA and circle scores (Fig. S4d) among the test molecules before and after Modof-pipe optimization (δ=0.4).

Fig. S4a shows that Modof-pipe is able to significantly improve plogP (mean value changed from -2.75 to 2.25 due to the Modof-pipe optimization). Among the improved plogP values, improvement from logP contributes most as Fig. S4b shows, where logP mean values changed from -1.40 to 2.85. Even though, SA scores are also improved by Modof-pipe

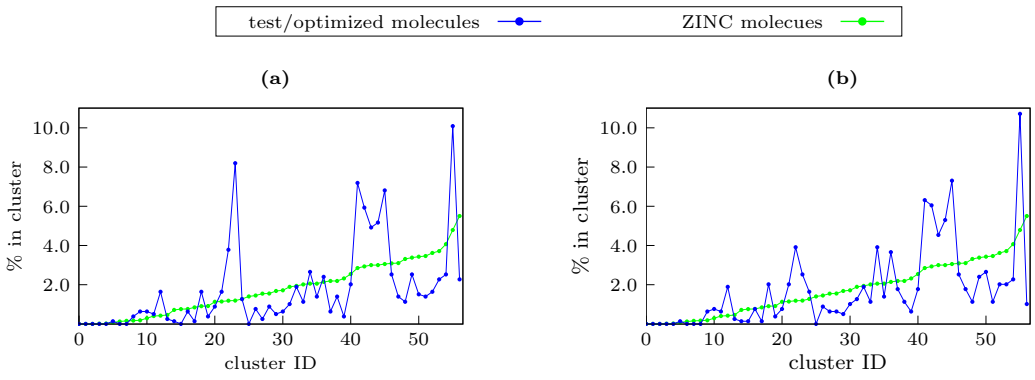


Fig. S5 | Molecule Transformation via Modof-pipe optimization in ZINC Chemical Space. a, Test molecules representativeness. **b,** Optimized molecules representativeness.

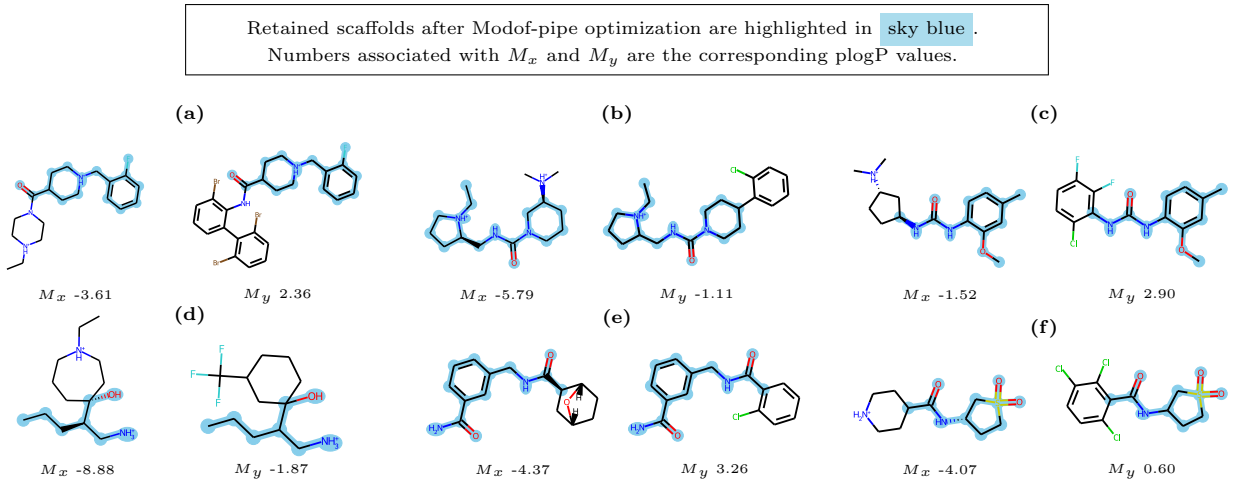


Fig. S6 | Examples of Test Molecules and their Optimized Molecules in plogP in Different ZINC Subspaces.

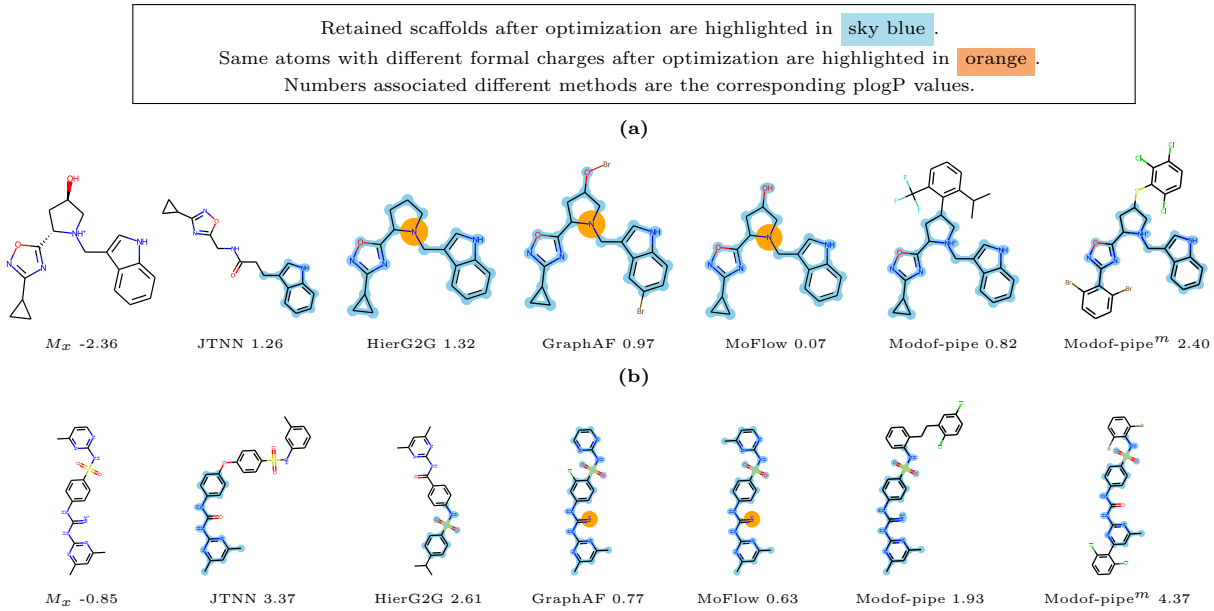


Fig. S7 | Examples of Optimized Molecules in plogP by Different Methods

as in Fig. S4c, where the mean value of SA scores is improved from -1.30 to -0.64. This indicates that while focusing on learning how to improve logP via fragment-based modification, Modof-pipe is also able to learn and incorporate synthetic accessibility into its modification process, and generate new molecules that are even better synthesizable. Fig. S4d shows the improvement of SA scores and circle scores together. Note that circle scores are small if there are large rings, which are not preferable for synthesis. The improvement of SA and circle scores together also indicates that Modof-pipe modifies molecules into more synthesizable ones.

S9.5 Chemical Transformation via Modof-pipe Optimization

An important aspect of chemical optimization is to search for and explore a novel chemical subspace where molecules have better properties[59]. To analyze whether the optimized molecules given by Modof-pipe reside in a novel chemical subspace compared to the subspace of those before optimization, we conducted the following analysis. We clustered the following five groups of molecules all together: (1) the molecules in training pairs that have bad properties, denoted as $\{M_x\}_{\text{trn}}$; (2) the molecules in training pairs that have good properties, denoted as $\{M_y\}_{\text{trn}}$; (3) the molecules in the benchmark test dataset, denoted as $\{M_x\}_{\text{tst}}$ ($|\{M_x\}_{\text{tst}}| = 800$); (4) the optimized test molecules by Modof-pipe with similarity constraint δ=0.4, denoted as $\{M_y\}_{\text{tst}}$; and (5) the rest, all the ZINC dataset molecules that are not in the above four groups. We did the clustering in a same way as in Section S3.1 using CLUTO. The clustering results represent how the ZINC chemical space looks, and how molecules are distributed in the ZINC chemical space.

Fig. S5 presents the results from 56 clusters that are sorted in the same way as in Section S3.1. Fig. S5a and Fig. S5b show that the test molecules and their optimized ones have roughly similar distributions over ZINC clusters (e.g., more distributed in cluster 40 - 50). This indicates that Modof-pipe optimization is not biased at a certain chemical subspace. On the other hand, the distributions still have some notable difference, for example, before optimization, about 8% of test molecules were in cluster 23; after optimization, only 4% of the optimized molecules are in cluster 23. Actually, among the 800 test molecules, 24.38% of the molecules changed to different clusters after Modof-pipe optimization. Compared to test molecules, among the training data, only about 9.20% of $\{M_x\}_{\text{trn}}$ have their paired $\{M_y\}_{\text{trn}}$ in a different cluster. Fig. S6

Table S4 | Model Complexity for plogP Optimization

model	number of parameters
JT-VAE	4.53M
GCPN	6.00M
JTNN (δ=0.4)	3.74M
JTNN (δ=0.6)	3.15M
HierG2G	6.80M
GraphAF	1.86M
MoFlow	130.35M
Modof (δ=0.0)	0.56M
Modof (δ=0.2)	0.54M
Modof (δ=0.4)	1.89M
Modof (δ=0.6)	0.51M

Table S5 | Hyper-Parameter Space for plogP Optimization

Hyper-parameters	Space
hidden layer dimension	{32, 64, 128, 256}
atom/node embedding dimension	{32, 64, 128, 256}
$\mathbf{z}^+/\mathbf{z}^-$ dimension	{8, 16, 32}
# iterations of GMPN	{3, 4, 5, 6}
# iterations of TMPN	{2, 3, 4}
# sampling	20

shows the examples of test molecules for which their optimized molecules changed to different clusters. The analysis results indicate that Modof-pipe is able to learn from training data (training chemical subspace), and also explore and transform into novel chemical subspaces. Note that Modof-pipe always constrains that the optimized molecules are at least δ-similar to those before optimization (in this analysis, δ=0.4). Under this constraint, the fact that Modof-pipe is able to find novel chemical subspaces for about 24% of test molecules indicates its strong exploration capability over the entire chemical space for molecule optimization.

S9.6 Comparison on Retaining Molecule Scaffolds

Fig. S7 presents two examples of molecules optimized by the four baseline methods, Modof-pipe and Modof-pipe^m, with similarity constraint δ=0.4. In these two examples, JTNN tends to insert a new structure within the middle of the molecules. HierG2G could have either very minor (in Fig. S7a) or very dramatic (Fig. S7b) change to the major molecular scaffolds. Both GraphAF and MoFlow tend to have very small modifications and therefore plogP improvement from these two methods is also small. Modof-pipe and Modof-pipe^m tend to change the periphery of the molecules but can guarantee to retain the major scaffolds of the molecules, with substantial plogP improvement.

S9.7 Model Complexity Comparison

Table S4 presents the number of parameters of baselines and our models. As shown in this table, the optimal Modof has 0.56M parameters with $\delta=0.0$, 0.54M parameters with $\delta=0.2$, 1.89M parameters with $\delta=0.4$ and 0.51M parameters with $\delta=0.6$, which are far less than those in the best baselines. For example, JTNN has 3.74M parameters with $\delta=0.4$ and 3.15M with $\delta=0.6$, and HierG2G has 6.8M parameters, that is, Modof uses at least 40% fewer parameters and 26% less training data but outperforms or achieves very comparable results as these state-of-the-art baselines.

S9.8 Parameters for Reproducibility of plogP Optimization

We implemented our models using Python-3.6.9, Pytorch-1.3.1, RDKit-2019.03.4 and NetworkX-2.3. We trained the models using a Tesla P100 GPU and a CPU with 16GB memory on Red Hat Enterprise 7.7. We tuned the hyper-parameters of our models with the grid-search algorithm in the parameter space presented in Table S5. We determined the optimal hyper-parameters according to their corresponding plogP property improvement over the validation molecules. To optimize a molecule, we randomly sampled K=20 latent vectors in each Modof iteration.

For δ=0.0, the optimal dimension of all the hidden layers is 128, and the dimension of latent embedding $\mathbf{z}$ is 16 (i.e., 8 for $\mathbf{z}^-$ and $\mathbf{z}^+$, respectively), in Modof. The optimal iterations of graph message passing GMPN and tree message passing TMPNare 6 and 4, respectively. For δ=0.2, the optimal dimension of all the hidden layers is 128, and the dimension of latent embedding $\mathbf{z}$ is 32, and the optimal iterations of GMPN and TMPN are 5 and 4, respectively. For δ=0.4, the optimal dimension of all the hidden layers is 256, and the dimension of latent embedding $\mathbf{z}$ is 32, and the optimal iterations of GMPN and TMPN are 6 and 3, respectively. For δ=0.6, the optimal dimension of all the hidden layers is 128, the dimension of latent embedding $\mathbf{z}$ is 32, and the optimal iterations of GMPN and TMPN are 4 and 3, respectively.

We optimized the models with learning rate 0.001 and batch size 32. During the training period, we did not use regularization and dropout and used default random number seeds in Pytorch to sample the noise variables employed in the reparameterization trick of VAE. The best performance was typically achieved within 7 epochs of training. We set the KL regularization weight β in the loss function (Equation 11 in the main manuscript) as 0.1 in the first epoch, and increased its value by 0.05 every 500 batches until 0.5.

S10 Fragment and Molecule Size Analysis in Training Data for plogP Optimization

Among the training molecules for plogP optimization, the top-5 most popular fragments that have been removed from M_x are: O[C:1] (6.43%), N#[C:1] (4.56%), [O-][C:1] (3.23%), [NH3+][C:1] (2.44%), N[C:1]=O (2.17%); the top-5 most popular fragments to be attached into M_y are: CCSc1cccc[c:1]1 (13.92%), Clc1ccc([C:1])cc1 (12.12%), Clc1c[c:1]ccc1

Table S6 | Data Statistics for QED and DRD2 Optimization

description	DRD2	QED
#training molecules	59,696	83,161
#training (M_x, M_y) pairs	125,469	92,045
#validation molecules	500	360
#test molecules	1,000	800
average similarity of training (M_x, M_y) pairs	0.6803	0.6578
average pairwise similarity between training and test molecules	0.1366	0.1227
average training molecule size	28.76	23.22
average training $\{M_x\}$ size	28.81	24.38
average training $\{M_y\}$ size	28.66	22.06
average test molecule size	24.57	22.81
average $\{M_x\}$ score	0.1827	0.7500
average $\{M_y\}$ score	0.5366	0.8768
average testing molecule score	0.0067	0.7528
average score improvement in training (M_x, M_y) pairs	0.4013	0.1355

(6.24%), Clc1cccc[c:1]1 (4.67%), c1ccc2sc([C:1])nc2c1 (4.00%). These removal and attaching fragments are visualized in Fig. 1a and Fig. 1b in the main manuscript.

Overall, the removal fragments in training data are on average of 2.85 atoms and the new attached fragments are of 7.55 atoms. In addition, 39.48% M_x molecules do not have fragments removed and only have new attached fragments, while only 1.78% M_x molecules do not have new fragments attached and only have fragments removed. This shows that in training data, the optimization of plogP is typically done via removing small fragments and then attaching larger fragments. This is also reflected in Table S3 ("$\#a_x$", "$\#a_y$") that out of each Modof iteration, the optimized molecules become larger. We observed the similar trend from JT-VAE and JTNN that their optimized molecules are also larger than those before optimization. In the benchmark data, larger molecules typically have better plogP values (e.g., the correlation between molecule size and plogP values is 0.42).

S11 Experimental Results on DRD2 and QED Optimization

In addition to improving plogP, another two popular benchmarking tasks for molecule optimization include improving molecule binding affinities against the dopamine D2 receptor (DRD2), and improving the drug-likeness estimated by quantitative measures (QED)[33]. Specifically, given a molecule that doesn't bind well to the DRD2 receptor (e.g., with low binding affinities), the objective of optimizing DRD2 property is to modify the molecule into another one that will better bind to DRD2. How well the molecules bind to DRD2 is assessed by a support vector machine classifier developed by Olivecrona *et al.*[34], which predicts a DRD2 score to measure the binding. In the QED task, given a molecule that is not much drug-like, the objective of optimizing QED property is to modify this molecule into a more "drug-like" molecule. The drug-likeness of molecules is quantified by comparing them with approved drugs on eight widely used molecular properties such as the number of aromatic rings and molecular polar surface area[33]. Note that for these two tasks, Modof does not restrict the size of the optimized molecules.

S11.1 Training Data Generation for DRD2 and QED Optimization

We used the CheMBL dataset[60] processed by Olivecrona *et al.*[34]. This processed CheMBL dataset has 1,179,477 molecules in total, and each molecule is restricted to have 10 to 50 heavy atoms and only contains atoms in {H, B, C, N, O, F, Si, P, S, Cl, Br, I}. We constructed our training data from this processed CheMBL dataset as follows. We first identified 7,743,098 pairs of molecules with similarities $\text{sim}(M_x, M_y) \geq 0.6$ from this CheMBL dataset. We selected this high similarity threshold because Modof requires training pairs different in only one fragment at one disconnection site, and thus should be very similar. Among these similar molecule pairs, we selected the pairs that satisfy the following property constraints, respectively: for the DRD2 optimization task, the DRD2 score difference of the two molecules in a pair should be at least 0.2, that is,

$$\text{DRD2}(M_y) - \text{DRD2}(M_x) \geq 0.2; \tag{S1}$$

for the QED optimization task, the QED score difference of the two molecules in a pair should be at least 0.1, that is,

$$\text{QED}(M_y) - \text{QED}(M_x) \geq 0.1. \tag{S2}$$

Among the molecule pairs that satisfied the above property constraints, respectively, we identified the pairs in which the two molecules are different only at one disconnection site. Through all these processes, we finally identified 125,469 training pairs for the DRD2 task and 92,045 training pairs for the QED task. The test set for the DRD2 and QED tasks is the benchmark datasets provided by Jin *et al.*[14] and contains 1,000 molecules and 800 molecules, respectively. Table S6 presents the data statistics for the DRD2 and QED tasks.

Fragment and Molecule Size Analysis for DRD2 and QED Optimization

Among the training molecules for the DRD2 optimization task, the top-5 most popular fragments that have been removed from M_x are: CO[C3:1] (2.90%), C[C3:1] (2.29%), C1C[N:1]CCO1 (1.67%), c1cc[cH:1]cc1 (1.63%), O=[N+]([O-])[C:1] (1.61%); the top-5 most popular fragments to be attached into M_y are: Cl[C:1] (4.21%) F[C:1] (4.13%) C[C:1] (1.95%) c1cc[c:1]cc1 (1.89%) O[C:1] (1.59%). Overall, the attaching fragments are on average of 6.46 atoms and the removed fragments are of 6.51 atoms. In addition, 7.08% M_x molecules do not have new fragments attached and only have fragments removed, while 9.20% M_x do not have fragments removed and only have fragments attached. Unlike in plogP

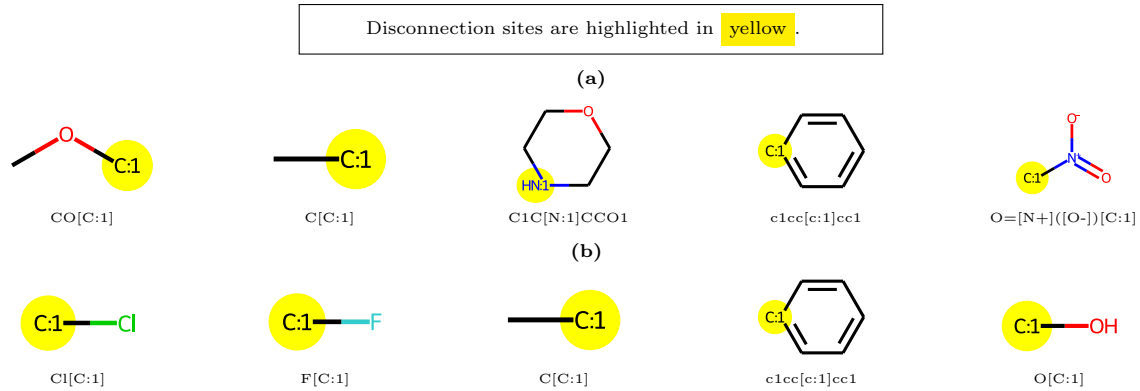


Fig. S8 | Popular Fragments in DRD2 Training Data. a, Visualization of popular removal fragments in DRD2 training molecules. **b,** Visualization of popular attaching fragments in DRD2 training molecules.

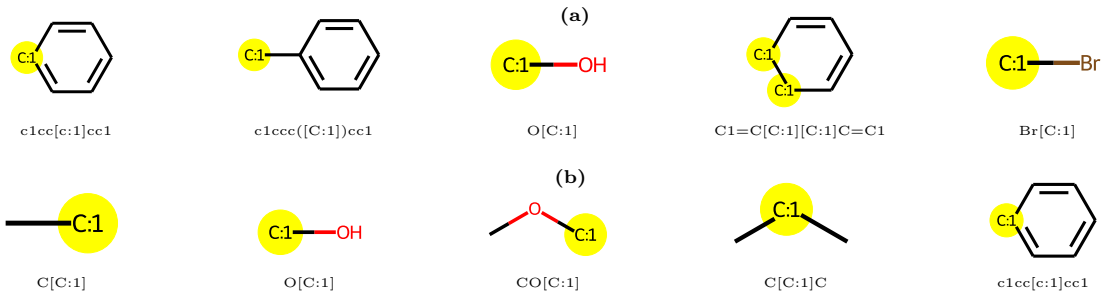


Fig. S9 | Popular Fragments in QED Training Data. a, Visualization of popular removal fragments in QED training molecules. **b,** Visualization of popular attaching fragments in QED training molecules.

training data, molecules in DRD2 training data do not have significant difference in size when they have different DRD2 properties. Fig. S8 presents the popular removal and attaching fragments among DRD2 training molecules.

Among the training molecules for the QED optimization task, the top-5 most popular fragments that have been removed from M_x are: c1cc[c:1]cc1 (4.04%), c1ccc([C:1])cc1 (2.52%), O[C:1] (2.20%), C1=C[C:1][C:1]C=C1 (2.03%), Br[C:1] (1.44%); the top-5 most popular fragments to be attached into M_y are: C[C:1] (5.10%), O[C:1] (3.40%), CO[C:1] (2.69%), C[C:1]C (2.57%), c1cc[c:1]cc1 (2.55%). Overall, the attaching fragments are on average of 4.17 atoms and the removed fragments are of 6.78 atoms. In addition, 17.62% M_x molecules do not have new fragments attached and only have fragments removed, while only 5.02% M_x molecules do not have fragments removed and only have fragments attached. Unlike in plogP and DRD2 training data, molecules in QED training data tend to be smaller when they have better QED properties. Fig. S9 presents the popular removal and attaching fragments among QED training molecules.

S11.2 Baseline Methods

We used JTNN[14] and HierG2G[15] as the baseline methods to compare Modof with. These two methods have been demonstrated to achieve the state-of-the-art performance on DRD2 and QED optimization[15]. Note that in the original JTNN and HierG2G manuscripts, their training data are different from the training data we generated as in Section S11.1: in their training data, molecule pairs have a similarity difference not below 0.4 ($\text{sim}(M_x, M_y) \geq 0.4$); for the DRD2 task, the DRD2 score of molecule M_x is in $(0, 0.05]$ and M_y in $[0.5, 1)$; for the QED task, the QED score of molecule M_x is in $[0.7, 0.8]$ and M_y to be $[0.9, 1)$. To allow a fair comparison, we also trained JTNN and HierG2G on Modof's training data, and the corresponding models are denoted as JTNN(m) and HierG2G(m), respectively.

Same as in the plogP task, in each iteration, Modof samples 20 times from its latent space and decodes 20 output molecules. In Modof-pipe, the best molecule which satisfies the corresponding similarity constraint and has positive improvement on DRD2 or QED at each iteration will be fed into the next iteration for further optimization. Each test molecule will be optimized with at most five iterations. In addition to Modof-pipe, Modof-pipe^m is also applied to enable multiple optimized molecules as output.

S11.3 Evaluation Metrics

We compared different methods by analyzing the following three groups of molecules:

- Optimized molecules (OM): If an input molecule M_x is optimized into M_y with any better properties, that is, no constraints on how much better M_y is than M_x, and M_y also satisfies the similarity constraint with M_x (i.e., $\text{sim}(M_y - M_x) > \delta$), this molecule is considered as optimized.

- OM under property improvement constraints (OM-pic): We also measured the optimized molecules (OM defined as above) that achieved a certain property improvement, following JTNN and HierG2G's criteria:

Table S7 | Overall Comparison on Optimizing DRD2

δ	model	OM			OM-pic (DRD2(M_y) ≥ 0.5)			OM-trn (imprv ≥ 0.2)		
		rate%	imprv±std	sim±std	rate%	imprv±std	sim±std	rate%	imprv±std	sim±std
0.4	JTNN	78.90	0.83±0.18	0.44±0.05	78.10	0.83±0.17	0.44±0.05	78.30	0.83±0.17	0.44±0.05
	HierG2G	85.40	0.81±0.20	0.44±0.05	82.00	0.83±0.16	0.44±0.05	84.00	0.82±0.18	0.44±0.05
	JTNN(m)	84.40	0.49±0.32	0.49±0.08	43.50	0.77±0.15	0.49±0.08	61.60	0.65±0.24	0.49±0.08
	HierG2G(m)	91.50	0.53±0.32	0.49±0.08	51.80	0.78±0.15	0.49±0.08	70.20	0.66±0.24	0.49±0.08
	Modof-pipe	98.80	0.69±0.29	0.48±0.07	74.90	0.83±0.14	0.48±0.07	89.00	0.75±0.22	0.48±0.07
	Modof-pipe^m	**99.10**	0.82±0.22	0.46±0.05	**88.60**	0.88±0.12	0.46±0.05	**95.90**	0.84±0.18	0.46±0.05
0.5	JTNN	13.40	0.73±0.23	0.54±0.05	12.50	0.78±0.17	0.54±0.05	12.60	0.77±0.17	0.54±0.04
	HierG2G	22.40	0.63±0.32	0.55±0.05	16.20	0.80±0.16	0.54±0.05	18.50	0.75±0.21	0.54±0.05
	JTNN(m)	64.90	0.35±0.32	0.57±0.07	22.50	0.74±0.15	0.57±0.07	35.80	0.59±0.23	0.57±0.06
	HierG2G(m)	78.10	0.39±0.33	0.57±0.07	30.80	0.75±0.15	0.56±0.06	45.80	0.61±0.24	0.57±0.06
	Modof-pipe	94.20	0.49±0.34	0.56±0.05	47.70	0.79±0.15	0.56±0.05	66.80	0.66±0.24	0.56±0.05
	Modof-pipe^m	**97.80**	0.63±0.32	0.54±0.04	**66.60**	0.82±0.14	0.54±0.04	**83.20**	0.72±0.23	0.54±0.04
0.6	JTNN	1.60	0.69±0.25	0.65±0.03	1.40	0.76±0.16	0.64±0.03	1.40	0.76±0.16	0.64±0.03
	HierG2G	4.40	0.50±0.35	0.65±0.04	2.60	0.77±0.15	0.64±0.04	3.10	0.70±0.21	0.65±0.04
	JTNN(m)	41.40	0.23±0.28	0.67±0.06	8.40	0.72±0.15	0.66±0.06	15.50	0.54±0.23	0.66±0.05
	HierG2G(m)	52.00	0.24±0.28	0.66±0.06	10.50	0.72±0.14	0.66±0.05	19.70	0.54±0.23	0.66±0.05
	Modof-pipe	79.60	0.27±0.31	0.65±0.04	19.70	0.75±0.15	0.64±0.03	32.40	0.59±0.24	0.64±0.03
	Modof-pipe-5	**87.50**	0.34±0.33	0.64±0.04	**28.90**	0.76±0.15	0.64±0.03	**45.30**	0.61±0.24	0.63±0.03

Columns represent: "rate%": the percentage of optimized molecules in each group (OM, OM-pic, OM-trn) over all test molecules; "imprv": the average property improvement; "std": the standard deviation; "sim": the similarity between the original molecules M_x and optimized molecules M_y. Best rate% values are in **bold.**

(1) For DRD2, the optimized molecules M_y should have DRD2 score no less than 0.5, that is,

$$\text{DRD2}(M_y) \geq 0.5; \tag{S3}$$

(2) For QED, the optimized molecules M_y should have QED score no less than 0.9, that is,

$$\text{QED}(M_y) \geq 0.9. \tag{S4}$$

- OM following training data property constraints (OM-trn): a potential issue with OM-pic is that if in the training molecule pairs, the property improvement constraints are not satisfied, imposing the constraints in OM for evaluation is overkilling and tends to underestimate the model performance. Therefore, we also measured the optimized molecules (OM defined as above) that achieved property improvement in a similar degree as in training data:

(1) For DRD2, the optimized molecules M_y should have DRD2 scores such that

$$\text{DRD2}(M_y) - \text{DRD2}(M_x) \geq 0.2 \tag{S5}$$

that is, same criterion as in training data generation (Equation S1).

(2) For QED, the optimized molecules M_y should have QED scores such that

$$\text{QED}(M_y) - \text{QED}(M_x) \geq 0.1, \tag{S6}$$

that is, same criterion as in training data generation (Equation S2).

Among each of these three groups of molecules, we measured 1) the percentage of the optimized molecules from that group over all test molecules, referred to as success rate, denoted as rate%, 2) within each group, the average property improvement of optimized M_y over M_x, and 3) within each group, the average similarity values between M_x and M_y. Note that rate% in OM-pic is identical to the success rate used in Jin *et al.*[14](JTNN) and Jin *et al.*[15](HierG2G).

S11.4 Experimental Results

Experimental Results on Optimizing DRD2

Table S7 presents the overall comparison between the baseline methods JTNN, HierG2G, JTNN(m), HierG2G(m) and our methods Modof-pipe and Modof-pipe^m on optimizing the DRD2 property. Please note that JTNN and HierG2G used a training dataset generated based on their criteria[14]; JTNN(m), HierG2G(m), Modof-pipe and Modof-pipe^m used the training data generated as described in Section S11.1; all the methods were tested on the same benchmark test set provided by Jin *et al.*[14] that contains 1,000 molecules. We stratified JTNN, JTNN(m), HierG2G and HierG2G(m)'s optimized molecules M_y's using their similarities with respect to corresponding M_x's. We used the similarity thresholds δ=0.4, 0.5 and 0.6 for the stratification, which also define the similarity constraints in Modof-pipe and Modof-pipe^m. For Modof-pipe^m, the property improvement and optimized molecule similarities to those before optimization are always considered on the best optimized molecule among all the optimized molecules. Under each similarity constraint δ, the optimized molecules within each of the OM, OM-pic and OM-trn groups are always at least δ-similar to the molecules before optimization.

Table S7 shows that Modof-pipe^m consistently achieves the highest success rates of optimized molecules, when considering molecules that are ever optimized (in OM), or optimized with a certain property improvement (in OM-pic and OM-trn), under all the similarity constraints (δ=0.4, 0.5 and 0.6). With similarity constraint δ=0.4, that is, each optimized molecule M_y has a similarity at least 0.4 with its corresponding M_x before optimization, Modof-pipe^m is able

Table S8 | Overall Comparison on Optimizing QED

δ	model	OM			OM-pic (QED(M_y) $\geq$0.9)			OM-trn (imprv$\geq$0.1)		
		rate%	imprv±std	sim±std	rate%	imprv±std	sim±std	rate%	imprv±std	sim±std
0.4	JTNN	71.00	0.16±0.04	0.47±0.06	60.50	0.17±0.03	0.47±0.06	67.38	0.17±0.03	0.47±0.07
	HierG2G	86.50	0.17±0.04	0.46±0.06	**75.12**	0.18±0.03	0.46±0.06	82.38	0.17±0.03	0.46±0.06
	JTNN(m)	93.50	0.13±0.06	0.55±0.10	40.50	0.17±0.03	0.54±0.09	68.50	0.15±0.03	0.54±0.09
	HierG2G(m)	91.75	0.13±0.06	0.52±0.10	37.12	0.17±0.03	0.52±0.09	65.88	0.15±0.03	0.53±0.10
	Modof-pipe	96.38	0.13±0.05	0.52±0.09	40.00	0.17±0.03	0.51±0.08	70.00	0.16±0.03	0.51±0.08
	Modof-pipe-5	**99.12**	0.15±0.04	0.48±0.07	66.25	0.18±0.03	0.48±0.07	**87.62**	0.17±0.03	0.48±0.07
0.5	JTNN	42.38	0.15±0.04	0.56±0.05	30.25	0.17±0.03	0.55±0.05	37.88	0.16±0.03	0.56±0.05
	HierG2G	55.00	0.16±0.04	0.55±0.05	40.38	0.17±0.03	0.55±0.05	50.38	0.16±0.03	0.55±0.05
	JTNN(m)	86.62	0.12±0.06	0.60±0.08	30.50	0.17±0.03	0.60±0.07	56.12	0.15±0.03	0.60±0.07
	HierG2G(m)	84.25	0.11±0.06	0.59±0.08	26.75	0.17±0.03	0.59±0.06	53.25	0.15±0.03	0.60±0.07
	Modof-pipe	89.25	0.12±0.05	0.59±0.07	27.25	0.17±0.03	0.58±0.06	53.62	0.15±0.03	0.58±0.06
	Modof-pipe-5	**98.62**	0.13±0.05	0.56±0.06	**43.38**	0.17±0.03	0.56±0.05	**71.25**	0.16±0.03	0.56±0.05
0.6	JTNN	17.62	0.13±0.05	0.65±0.05	10.12	0.16±0.03	0.65±0.04	13.38	0.15±0.03	0.65±0.04
	HierG2G	20.25	0.14±0.06	0.65±0.05	12.00	0.17±0.03	0.65±0.04	15.75	0.16±0.03	0.65±0.04
	JTNN(m)	73.62	0.10±0.07	0.68±0.07	**18.88**	0.17±0.03	0.66±0.06	38.62	0.15±0.03	0.66±0.05
	HierG2G(m)	70.88	0.10±0.07	0.67±0.06	17.38	0.17±0.03	0.66±0.05	37.25	0.15±0.03	0.66±0.05
	Modof-pipe	66.25	0.09±0.06	0.66±0.05	12.25	0.16±0.03	0.65±0.04	29.62	0.14±0.03	0.65±0.04
	Modof-pipe-5	**89.25**	0.09±0.07	0.66±0.05	18.62	0.17±0.03	0.65±0.04	**39.50**	0.15±0.03	0.65±0.04

Columns represent: "rate%": the percentage of optimized molecules in each group (OM, OM-pic, OM-trn) over all test molecules; "imprv": the average property improvement; "std": the standard deviation; "sim": the similarity between the original molecules M_x and optimized molecules M_y. Best rate% values are in **bold.**

to improve the DRD2 property for 99.10% of all the test molecules in OM; this is a 8.31% improvement over the best baseline method HierG2G(m), which achieves a 91.5% success rate. Among the optimized molecules in OM, the average DRD2 property improvement ("imprv±std") from Modof-pipe^m (0.82±0.22) is among the best compared to the results from other baseline methods (e.g., 0.83±0.18 from JTNN); the average pair-wise similarities between the test molecules and their optimized molecules from Modof-pipe^m (0.46±0.05) are also comparable to those of other methods. Compared to OM with δ=0.4, the optimized molecules in OM-pic and OM-trn have a lower success rate because not all the optimized molecules in OM have DRD2(M_y) $\geq$ 0.5 or DRD2 improvement no less than 0.2. However, Modof-pipe^m still achieves the best success rates (88.60% in OM-pic and 95.90% in OM-trn, respectively) compared to the baseline methods. In terms of the property improvement among the optimized molecules in OM-pic, Modof-pipe^m achieves the best (0.88±0.12); in terms of pair-wise similarities, Modof-pipe^m has very comparable performance (0.46±0.05) compared to the baseline methods. The same conclusions hold true in OM-trn.

With similarity constraint δ=0.5 and 0.6, we observed the same trends as those with δ=0.4: consistently in OM, OM-pic and OM-trn, Modof-pipe^m achieves the best success rates (rate%), best or competitive property improvement (imprv±std), and competitive pair-wise similarities (sim±std) between the test molecules and their optimized molecules. Particularly, with δ=0.5 and 0.6, Modof-pipe^m's success rates in OM, OM-pic and OM-trn were substantially higher than those of the baseline methods: for example, with δ=0.5 and 0.6 in OM-pic, Modof-pipe^m has a 116.23% and 175.24%, respectively, higher rate% than that of HierG2G(m) (66.60% vs 30.80%; 28.90% vs 10.50%); in OM-trn, Modof-pipe^m's rate% is 81.66% and 129.95%, respectively, higher than that of HierG2G(m) (83.20% vs 45.80%; 45.30% vs 19.70%). This demonstrates the strong capability of Modof-pipe^m in optimizing molecules and improving DRD2 properties.

Experimental Results on Optimizing QED

Table S8 presents the overall comparison between the baseline methods JTNN, HierG2G, JTNN(m), HierG2G(m) and our methods Modof-pipe and Modof-pipe^m on optimizing the QED property. Overall, the trends are very similar with what we observed in DRD2 optimization: Modof-pipe^m achieves the best success rates in OM, OM-pic and OM-trn with all the similarity constraints (δ=0.4, 0.5 and 0.6), except in OM-pic with δ=0.4 and 0.6, where Modof-pipe^m has the second best success rates (with δ=0.6, the difference with the best success rate is very minimum).

Parameters for Reproducibility of Optimizing DRD2 and QED

We tuned the hyper-parameters of our models for DRD2 and QED optimization tasks with the grid-search algorithms in the parameter spaces as presented in Table S9. We determined the optimal hyper-parameters according to the success rate rate% of Modof-pipe in OM-pic under δ=0.4 over the validation molecules provided by Jin *et al.*[14]. In each iteration, we randomly sampled K=20 latent vectors and output 20 optimized candidates.

For DRD2, the optimal dimension of all the hidden layers is 320 and the dimension of latent embedding $\mathbf{z}$ is 64 (i.e., 32 for $\mathbf{z}^-$ and $\mathbf{z}^+$, respectively). The optimal iterations of graph message passing GMPN and tree message passing TMPN are 6 and 5, respectively. The optimal Modof for DRD2 has 3.09M parameters and can achieve the best performance with 3 epochs of training. For QED, the optimal dimension of all the hidden layers is 256 and the dimension of latent embedding $\mathbf{z}$ is 32 (i.e., 16 for $\mathbf{z}^-$ and $\mathbf{z}^+$, respectively). The optimal iterations of graph message passing GMPN and tree message passing TMPN is 4. The optimal Modof for QED has 1.81M parameters and can achieve the best performance with 4 epochs of training. Other training details such as learning rate and KL regularization weight are the same as demonstrated in Section S9.8.

S12 Experimental Results on Multi-Property Optimization of DRD2 and QED

Here we demonstrate the effectiveness of Modof for multi-property optimization. We conduct experiments to simultaneously improve two molecular properties (QED and DRD2). Specifically, given a molecule that is not much drug-like (i.e.,

Table S9 | Hyper-Parameter Space for DRD2 and QED Optimization

Hyper-parameters	Space
hidden layer dimension	{64, 128, 256, 320}
atom/node embedding dimension	{64, 128, 256, 320}
$\mathbf{z}^+/\mathbf{z}^-$ dimension	{8, 16, 32}
# iterations of GMPN	{4, 5, 6, 7}
# iterations of TMPN	{3, 4, 5}
# sampling	20

Table S10 | Data Statistics for Multi-Property Optimization of DRD2 and QED

description	DRD2 & QED
#training molecules	10,121
#training (M_x, M_y) pairs	14,230
#validation molecules	200
#test molecules	800
average similarity of training (M_x, M_y) pairs	0.6721
average pairwise similarity between training and test molecules	0.1257
average training molecule size	30.08
average training $\{M_x\}$ size	32.42
average training $\{M_y\}$ size	27.16
average test molecule size	32.05
average $\{$DRD2(M_x)$\}$ score	0.1788
average $\{$DRD2(M_y)$\}$ score	0.4957
average $\{$QED(M_x)$\}$ score	0.5033
average $\{$QED(M_y)$\}$ score	0.6963
average DRD2 score of test molecules	0.1102
average QED score of test molecules	0.4241
average DRD2 score improvement in training (M_x, M_y) pairs	0.4356
average QED score improvement in training (M_x, M_y) pairs	0.2240

with a low DRD2 score) and doesn't bind well to the DRD2 receptor (i.e., with a low QED score), the objective of the task is to modify this molecule to a drug-like molecule bound well to the DRD2 (i.e., with both high DRD2 and QED scores). The assessment of DRD2 and QED properties is as described in the Section S11.

S12.1 Training Data Generation for Multi-Property Optimization

We used the training molecule pairs for DRD2 task derived from CheMBL dataset as in Section S11.1 to construct the training data for this multi-property optimization task. Recall that each molecule pair (M_x, M_y) for the DRD2 task has similarity $\text{sim}(M_x, M_y) \geq 0.6$ and satisfies the following property constraint:

$$\text{DRD2}(M_y) - \text{DRD2}(M_x) \geq 0.2; \tag{S7}$$

Among these molecule pairs, we first selected the pairs that also have differences on QED scores, that is,

$$\text{QED}(M_x) < 0.6 \text{ AND } \text{QED}(M_y) \geq 0.6. \tag{S8}$$

Among the molecule pairs that satisfied the above two property constraints, we first identified the pairs in which two molecules are different only at one disconnection site. We finally identified 14,230 training pairs for this task. The test set for this task includes 800 molecules that do not appear in the training pairs and have the property $\text{DRD2}(M) < 0.5$ and $\text{QED}(M) < 0.6$. Table S10 presents the data statistics for this multi-property optimization task.

Fragment and Molecule Size Analysis for Multi-Property Optimization

Among the training molecules for the multi-property optimization task, the top-5 most popular fragments that have been removed from M_x are: O=[N+]([O-])[C3:1] (5.17%), c1cc[c:1]cc1 (1.91%), c1ccc([C3:1])cc1 (1.90%), c1ccc2c(c1)ccc[c:1]2 (1.25%), CO[C3:1] (1.15%); the top-5 most popular fragments to be attached into M_y are: F[C3:1] (5.14%), Cl[C3:1] (3.32%), C[C3:1] (2.99%), O[C3:1] (2.31%), c1cc[c:1]cc1 (2.14%). In addition, 16.75% M_x do not have new fragments attached and only have fragments removed, while only a few M_x do not have fragments removed and only have fragments attached. This is likely due to the fact that smaller molecules have better QED properties.

S12.2 Baseline Methods

We used the same baselines JTNN and HierG2G as in the DRD2 and QED tasks in Section S11. These two methods have been demonstrated to achieve the state-of-the-art performance on DRD2 and QED optimization, respectively, and thus are strong baselines on the multi-property optimization task on DRD2 and QED. Note that the original JTNN and HierG2G did not have the experiments on this multi-property optimization task in their manuscripts. Therefore, we can only train JTNN and HierG2G with our dataset, and the corresponding models are denoted as JTNN(m) and HierG2G(m), respectively.

Same as in the single-property optimization task, in each iteration, Modof sampled 20 times from its latent space and decodes 20 output molecules. At each iteration, Modof-pipe and Modof-pipe^m used the sum of DRD2 and QED scores to select the best optimized molecules under the corresponding similarity constraint. The selected molecules were fed into the next iteration for further optimization. Each test molecule was optimized in at most five iterations.

Table S11 | Overall Comparison on Multi-Property Optimization of DRD2 and QED

δ	model	OM-pic (DRD2 ≥ 0.5)			OM-pic (QED ≥ 0.6)			OM-pic (DRD2 ≥ 0.5 && QED ≥ 0.6)		
		rate%	imprv±std	sim±std	rate%	imprv±std	sim±std	rate%	imprv±std	sim±std
0.4	JTNN(m)	10.62	0.53±0.20	0.54±0.11	21.00	0.27±0.13	0.55±0.12	8.62	0.82±0.22	0.55±0.11
	HierG2G(m)	15.62	0.58±0.18	0.51±0.10	26.00	0.30±0.13	0.51±0.10	14.50	0.85±0.21	0.51±0.10
	Modof-pipe	48.88	0.63±0.17	0.51±0.10	54.00	0.32±0.13	0.49±0.08	24.62	0.92±0.20	0.49±0.08
	Modof-pipe^m	**67.75**	0.71±0.15	0.47±0.07	**82.25**	0.36±0.13	0.48±0.07	**38.88**	0.95±0.21	0.48±0.06
0.5	JTNN(m)	6.88	0.53±0.19	0.60±0.09	13.12	0.26±0.12	0.62±0.10	5.88	0.82±0.21	0.60±0.09
	HierG2G(m)	8.38	0.55±0.17	0.59±0.08	15.25	0.24±0.12	0.60±0.08	7.88	0.80±0.21	0.60±0.08
	Modof-pipe	41.25	0.58±0.17	0.59±0.07	44.75	0.27±0.12	0.59±0.07	16.62	0.82±0.20	0.59±0.07
	Modof-pipe^m	**58.38**	0.66±0.15	0.56±0.06	**69.25**	0.32±0.12	0.57±0.06	**27.25**	0.84±0.23	0.57±0.07
0.6	JTNN(m)	2.88	0.49±0.15	0.68±0.06	7.25	0.24±0.11	0.70±0.07	2.62	0.75±0.18	0.68±0.06
	HierG2G(m)	4.38	0.53±0.15	0.66±0.05	8.12	0.23±0.12	0.67±0.06	4.12	0.75±0.20	0.66±0.05
	Modof-pipe	26.88	0.53±0.16	0.67±0.06	29.12	0.24±0.11	0.67±0.05	9.12	0.75±0.19	0.67±0.05
	Modof-pipe^m	**34.12**	0.58±0.17	0.66±0.05	**43.88**	0.27±0.11	0.66±0.05	**13.00**	0.78±0.22	0.66±0.06

Columns represent: "OM-pic (DRD2 ≥ 0.5)": the optimized molecules that have a DRD2 score no less than 0.5; "OM-pic (QED ≥ 0.5)": the optimized molecules that have a QED score no less than 0.6; "OM-pic (DRD2 ≥ 0.5 && QED ≥ 0.6)": the optimized molecules that have a DRD2 score no less than 0.5 and at the same time a QED score no less than 0.6; "rate%": the percentage of optimized molecules in each group over all test molecules; "imprv": the average property improvement on DRD2 or QED for the group OM-pic (DRD2 ≥ 0.5) or OM-pic (QED ≥ 0.6), or the average of sum of property improvements on DRD2 and QED scores for the group OM-pic (DRD2 ≥ 0.5 && QED ≥ 0.6); "std": the standard deviation; "sim": the similarity between the original molecules M_x and optimized molecules M_y. Best rate% values are in **bold.**

S12.3 Evaluation Metrics

Similarly to that in Section S11.3, we compared different methods by analyzing OM-pic on different groups of molecules as follows in the test set with respect to different property constraints:

- OM-pic for DRD2: The optimized molecules M_y should have DRD2 score no less than 0.5, that is,

$$\text{DRD2}(M_y) \geq 0.5. \tag{S9}$$

- OM-pic for QED: The optimized molecules M_y should have QED score no less than 0.6, that is,

$$\text{QED}(M_y) \geq 0.6. \tag{S10}$$

- OM-pic for DRD2 and QED: The optimized molecules M_y should have DRD2 score no less than 0.5 and QED score no less than 0.6, that is,

$$\text{DRD2}(M_y) \geq 0.5 \;\&\&\; \text{QED}(M_y) \geq 0.6. \tag{S11}$$

Note that in this multi-property optimization task, molecules that have been optimized to have $\text{DRD2}(M_y) \geq 0.5$ or $\text{QED}(M_y) \geq 0.6$ may or may not satisfy the QED or DRD2 constraint. The group OM-pic for DRD2 or OM-pic for QED only includes the optimized molecules that satisfy the DRD2 or QED constraint alone. The group OM-pic for DRD2 and QED includes the optimized molecules that satisfy both the DRD2 and QED constraints. In other words, this group is the intersection of the above two groups. Among each of these three groups of molecules, we used the success rate rate%, the average property improvement of M_y over M_x, and the average similarity values between M_x and M_y to compare the different results.

S12.4 Experimental Results

Table S11 presents the overall comparison between the baseline methods JTNN(m) and HierG2G(m) and our methods Modof-pipe and Modof-pipe^m on the multi-property optimization problem. All the methods were trained for 30 epochs on the same training data as generated in Section S12.1. We then tuned the hyper-parameters and found the best model for each method using a validation set with 200 molecules. The selected best models were tested on the test set that contains 800 molecules. Following Section S11.4, we used the similarity thresholds δ=0.4, 0.5 and 0.6 to stratify the optimized molecules of all the methods.

Table S11 shows that Modof-pipe and Modof-pipe^m significantly outperform two baseline methods JTNN(m) and HierG2G(m) on the success rates of optimized molecules under all the similarity constraints (δ=0.4, 0.5 and 0.6). With similarity constraint δ=0.4, that is, each optimized molecule M_y has a similarity at least 0.4 with its corresponding M_x before optimization, Modof-pipe and Modof-pipe^m is able to achieve 24.62% and 38.88% success rates on the group OM-pic (DRD2 ≥ 0.5 && QED ≥ 0.6), respectively; these are 69.79% and 168.14% better than the best baseline method HierG2G(m) (14.50% in OM-pic (DRD2 ≥ 0.5 && QED ≥ 0.6)), respectively. With other similarity constraints δ=0.5 and 0.6, we observed the same trend that Modof-pipe and Modof-pipe^m consistently outperformed two baseline methods on the results of group OM-pic (DRD2 ≥ 0.5 && QED ≥ 0.6). This demonstrates the strong capability of Modof-pipe and Modof-pipe^m on the multi-property optimization problem.

In addition, Table S11 shows that Modof-pipe and Modof-pipe^m also outperformed the two baseline methods JTNN(m) and HierG2G(m) with a wide margin on the success rates in group OM-pic(DRD2 ≥ 0.5) and OM-pic(QED ≥ 0.6). With similarity constraint δ=0.4, Modof-pipe and Modof-pipe^m were able to achieve 54.00% and 82.25% success rates on the group OM-pic(QED ≥ 0.6), respectively; these are 107.69% and 216.34% better the best baseline method HierG2G(m) (26.00% in OM-pic (DRD2 ≥ 0.5 && QED ≥ 0.6)), respectively. This demonstrates that Modof-pipe and Modof-pipe^m have stronger capabilities to optimize molecules under the similarity constraint towards better properties. Compared to these rate% values on the group in single-property optimization as in Table S11 (54.00% for Modof-pipe and 82.25% for Modof-pipe^m on QED optimization), the smaller success rates in the multi-property optimization (24.62% for Modof-pipe and 38.88% for Modof-pipe^m) also indicate that multi-property optimization is a challenging problem.

Parameters for Reproducibility of Multi-Property Optimization

We tuned the hyper-parameters of our models with the grid-search algorithms in the parameter spaces as presented in Table S9. The optimal hyper-parameters were determined according to the success rates rate% of Modof-pipe in the group OM-pic (DRD2 $\geq$ 0.5 && QED $\geq$ 0.6) under δ=0.4 over the 200 validation molecules. The optimal dimension of all the hidden layers is 128 and the dimension of the latent embedding $\mathbf{z}$ is 64 (i.e., 32 for $\mathbf{z}^-$ and $\mathbf{z}^+$, respectively). The optimal iterations of graph message passing GMPN and tree message passing TMPN is 4. The optimal Modof model has 0.51M parameters and can achieve the best performance with 8 epochs of training.

S13 Additional Discussions

S13.1 Local Greedy Optimization

A limitation of Modof-pipe is that it employs a local greedy optimization strategy: in each iteration, the input molecules to Modof will be optimized to the best, and if the optimized molecules do not have better properties, they will not go through additional Modof iterations. Table S2 shows that at δ=0.4, about 4% of such molecules in total (under "#n%") stop in the middle of 5-iteration optimization. However, it is possible that such optimized molecules with declined properties might be further optimized in the later Modof iterations. Fig. 1d in the main manuscript shows an example of such molecules, where $M_x^{(2)}$ has worse properties than $M_x^{(1)}$ after the second iteration of optimization, due to the replacement of two chlorine atoms with one fluorine atom which decreases logP but slightly increases SA. However, $M_x^{(2)}$ can be further optimized into $M_x^{(3)}$ of better properties in the third iteration, due to the addition of a more hydrophobic chlorophenyl group. Note that in our experiments (Table S2), Modof-pipe always stops if no property improvement is observed. Instead, we can exhaustively optimize each molecule through the entire Modof-pipe and identify the global optimality. In Modof-pipe^m, the top results from each iteration, regardless whether their properties are improved or not, will be further optimized in the next iteration. Therefore, it mitigates the local optimal issue.

S13.2 Multi-Property Optimization

In addition to partition coefficient, there are a lot of factors that need to be considered in order to optimize a lead, or in general, to develop a molecule into a drug. For example, toxicity and synthesizability are another two important factors for a promising drug candidate; potency, metabolism, cell permeability and side effects are also important for a successful drug. Almost all the existing computational methods can only optimize one single property, or a few properties that can be simply (linearly) combined as one objective[61,62] (e.g., plogP is a linear combination of logP, synthesis accessibility and ring size; in our experiments presented in the Supplementary Information Section S12, it is a linear combination of DRD2 and QED scores). These methods can be used to optimize one property after another (i.e., in a sequential order) by training and applying one model for each property of interest. However, this sequential optimization is typically suboptimal as later optimization on one property could alter the property optimized earlier.

Unfortunately, to develop one computational model that can best optimize multiple properties simultaneously toward global optimality is non-trivial,[63,64] and represents a challenging future research direction for generative models. There are many reasons contributing to the difficulty of multi-property optimization for computational drug development. First of all, it could be not trivial to set up appropriate mathematical objective or loss functions for the machine learning process. Existing methods use simple combinations (e.g., linear) of multiple objective or loss functions, each with respect to one specific desired drug property. Such simple combinations (e.g., via arithmetic mean or geometric mean) may already impose biases on the multiple properties to be optimized. For example, in plogP, the implicit bias is that hydrophobicity and synthesis accessibility are independent, which may or may not be true,[22] since the estimated hydrophobicity (logP) tends to be larger on larger molecules but the synthesis accessibility score would be penalized by larger molecule size. Learning with respect to a biased objective or loss function may lead to incorrect solutions. Even with the loss function accurately provided, it may still be highly non-trivial to solve or approximate the underlying mathematical or machine learning problems given their combinatorial nature. It is very likely the loss function is non-convex and its components cannot be easily decoupled, and therefore, it is very challenging to utilize existing learning strategies (e.g., alternating minimization,[65] teacher's forcing[66]). Meanwhile, it is in general very challenging to develop new, effective learning algorithms for multi-objective optimization.[67] Limited training data is another barrier. To train a truly multi-property optimization model, which is very likely more complex than a single-property optimization model, molecules satisfying concurrent constraints on multiple properties are needed. This would lead to a small set of eligible molecules to train a complex model, and thus the model cannot be well trained.

S13.3 Target-Specific Molecule Optimization

Most of the existing molecule optimization methods focus on optimizing properties that are general for all the drug candidates, regardless of the protein targets that these drug candidates are desired to bind to. There are quite a few such common properties that successful drugs need to exhibit, such as high solubility, low toxicity and small size, which ensure the general usability of the existing molecule optimization methods. However, it is also important that molecule optimization can be tailored to a given protein target, pathway or disease. There exist some studies on molecule optimization with respect to Dopamine receptor D2 (DRD2)[14,15] and Epidermal Growth Factor Receptor (EGFR),[68] but they do not consider the target information in the modeling, and also are limited by a small number of training molecules. Our methods have demonstrated good performance on optimizing with respect to DRD2 (Supplementary Information Section S11.4), but the limitation is that they cannot be easily adapted to incorporate additional information from protein targets, or in general any information rather than molecule structures that could be helpful for drug development purposes (e.g., dose

response, toxicity profiles). To optimize a molecule with respect to a certain protein target via deep learning methods, new techniques are needed to deal with the target information (e.g., 3D structures of the binding pockets) and modify molecule function groups (and molecule 3D structures) accordingly, particularly when the training data (e.g., known, accurate binding conformations and binding pocket structures) are much less than what we have for general molecule property optimization.

S13.4 Optimization in Other Areas

In addition for drug development, the Modof framework could also be used for compounds or substance property optimization in other application areas, such as band gaps for solar cells, organic light emitting diodes and dyes,[69] melting or boiling points for volatiles, materials and plastics,[70] and solubility for batteries.[71] This could be done by replacing the properties and their measurements of molecules in the loss function of Modof with those of compounds or substances in the respective application areas. Given the fact that Modof is data-driven, learns only from pairs of molecules which have difference in the interested properties, and does not require any application or domain-specific knowledge, it is flexible enough to be adapted to other applications where compound or substance properties can be measured and compared.

S14 Algorithms of Modof

Algorithm S1 describes the encoding process of Modof. Modof-encoder takes a pair of molecules as input and encodes the difference of the molecules into vector $\mathbf{z}_{xy}$. Algorithm S2 describes the decoding process of Modof. Modof-decoder modifies M_x into M_y according to the latent difference vector $\mathbf{z}_{xy}$. Specifically, in the modification procedure, Modof-decoder first identifies the scaffolds of molecules which should be retained after the optimization, and removes fragments that are not in the scaffolds to get the intermediate representation. Modof-decoder then modifies the intermediate molecule representation M^* into M_y by sequentially attaching new nodes to $\mathcal{T}^*$ in a breadth-first order as in Algorithm S3. Algorithm S4 describes Modof-pipe optimization. Given a molecule M_x, similarity constraint δ, the maximal number of samplings K, and the maximum number of iterations allowed maxIters, our Modof-pipe iteratively optimizes M_x into a new molecule M_y with better property ($\text{plogP}(M_y) > \text{plogP}(M_x)$) under the similarity constraint ($\text{sim}(M_x, M_y) \geq \delta$). Algorithm S5 describes Modof-pipe^m optimization. Given a molecule M_x, similarity constraint δ, the maximal number of samplings K, the maximum number of iterations allowed maxIters, the maximum number of input molecules in each iteration m and the maximum number of output molecules of Modof-pipe^m, in the t-th iteration, our Modof-pipe^m optimizes each input molecule, denoted as $M_x^{(t)}(i)$, into K decoded molecules, denoted as $\{{M_y}^{(t)}(i,k)|k = 1, \cdots, K\}$ under the similarity constraint. Modof-pipe^m then selects no more than m unique best molecules from $\cup_i\{{M_y}^{(t)}(i,k)|k = 1, \cdots, K\}$ for further optimization in the next t+1 iteration. The best b molecules among all decoded molecules $\cup_t \cup_i \{{M_y}^{(t)}(i,k)|k = 1, \cdots, K\}$ will be returned as the final output of Modof-pipe^m.

Algorithm S1 Modof-encoder

Require: $M_x = (\mathcal{G}_x, \mathcal{T}_x)$, $M_y = (\mathcal{G}_y, \mathcal{T}_y)$, n_d

▷ Atom embedding
$\{\mathbf{a}_x\} = \mathsf{GMPN}(\mathcal{G}_x)$
$\{\mathbf{a}_y\} = \mathsf{GMPN}(\mathcal{G}_y)$
▷ Node embedding
$\{\mathbf{n}_x\} = \mathsf{TMPN}(\mathcal{T}_x, \{\mathbf{a}_x\})$
$\{\mathbf{n}_y\} = \mathsf{TMPN}(\mathcal{T}_y, \{\mathbf{a}_y\})$
▷ Difference embedding
$\mathbf{z}_{xy} = \mathsf{DE}(\mathcal{T}_x, \{\mathbf{n}_x\}, \mathcal{T}_y, \{\mathbf{n}_y\}, n_d)$
return $\mathbf{z}_{xy}$

Algorithm S2 Modof-decoder

Require: $M_x = (\mathcal{G}_x, \mathcal{T}_x)$, $\mathbf{z}_{xy} = [\mathbf{z}^+_{xy}, \mathbf{z}^-_{xy}]$

▷ Disconnection site prediction
$n_d = \mathsf{DSP}\ (\mathcal{T}_x, \mathbf{z}_{xy})$
▷ Removal fragment prediction
$n_r = \mathsf{RFP}(\{\mathbf{n}_u|e_{ub} \in \mathcal{V}_x\}, \mathbf{z}^-_{xy})$
▷ Intermediate representation
$M^* = \mathsf{IMR}(n_r, M_x)$
▷ New fragment attachment
$M_y = \mathsf{NFA}(M^*, n_d, \mathbf{z}^+_{xy})$
return M_y

Algorithm S3 Modof New Fragment Attacher NFA

Require: $M^* = (\mathcal{G}^*, \mathcal{T}^*)$, n_d, $\mathbf{z}_{xy}^+$

$t = 0$
$M^{*(0)} = M^*$
Q = emptyQueue()
Q.push(n_d)
while !Q.isEmpty() **do**
    $n^{*(t)} = Q$.pop()
    ▷ Child prediction
    while NFA-cp ($n^{*(t)}$, $\mathbf{z}_{xy}^+$,$M^{*(t)}$) **do**
        ▷ Node type prediction
        (n_c, $\mathbf{x}_c$)= NFA-ntp($n^{*(t)}, \mathbf{z}_{xy}^+, M^{*(t)}$)
        ▷ Attachment point prediction
        (a_p^*, a_c^*) = NFA-app($n^{*(t)}, n_c, \mathcal{G}^{*(t)}, \mathbf{z}_{xy}^+$)
        ▷ Attach n_c to $M^{*(t)}$ at $n^{*(t)}$
        $M^{*(t)}$ = attach($M^{*(t)}$, $n^{*(t)}$, a_p^*, n_c, a_c^*)
        ▷ Breath-first child expansion
        Q.push(n_c)
    end while
    $M^{*(t+1)} = M^{*(t)}$
    $t = t + 1$
end while
return $M^{*(t)}$

Algorithm S4 Molecule Optimization via Modof-pipe

Require: M_x, δ, K, maxIters, property

$M_y^{(0)} = M_x$
for $t = 1$ to maxIters **do**
    ▷ input molecule to the t-th Modof module
    $M_x^{(t)} = M_y^{(t-1)}$
    ▷ best decoded molecule from this module
    $M_y^{\star(t)} = M_x^{(t)}$
    ▷ multiple samplings and decoding
    for $k = 0$ to K **do**
        $\mathbf{z}^{(k)}$ = sample from $\mathbf{z}$
        M = Modof-decoder($M_x^{(t)}, \mathbf{z}^{(k)}$)
        ▷ the best decoded molecule under constraints
        if property(M)>property($M_y^{\star(t)}$) and
            sim(M, M_x)$\geq\delta$ **then**
            $M_y^{\star(t)} = M$
        end if
    end for
    if isSame($M_y^{\star(t)}, M_x^{(t)}$) **then**
        ▷ no more optimization and Modof-pipe stops
        $t = t - 1$
        break
    else
        ▷ output molecule from the t-th module
        $M_y^{(t)} = M_y^{\star(t)}$
    end if
end for
$M_y = M_y^{(t)}$
return M_y

Algorithm S5 Molecule Optimization via Modof-pipe^m

Require: M_x, δ, K, maxIters, m, b

$\{M_y^{\star(0)}\} = \{M_x\}$
for $t = 1$ to maxIters **do**
  ▷ input m molecules to the t-th Modof module
  $\{M_x^{(t)}(i)|i = 1, ..., m\} = \{M_y^{\star(t-1)}\}$
  for $i = 0$ to m-1 **do**
    ▷ multiple samplings and decoding
    for $k = 0$ to K **do**
      sample $\mathbf{z}$ from latent space
      $M_y^{(t)}(i,k)$=Modof-decoder$(M_x^{(t)}(i), \mathbf{z})$
      ▷ the decoded molecule is unique and similar
      if sim$(M_y^{(t)}(i,k), M_x) \geq \delta$ **then**
        add $M_y^{(t)}(i,k)$ into $\{M_y^{(t)}(i,k)\}$
      end if
    end for
  end for
  ▷ select top-m molecules from $\cup_i \{M_y{}^{(t)}(i,k)\}$
  $\{M_y^{\star(t)}\}$=top$(\cup_i \{M_y{}^{(t)}(i,k)\}, m)$
end for
▷ select b best molecules from $\cup_t \cup_i \{M_y^{(t)}(i,k)\}$
$\{M_y^{\star}\}$=top$(\cup_t \cup_i \{M_y^{(t)}(i,k)\}, b)$
return $\{M_y^{\star}\}$

References

1. Jorgensen, W. L. Efficient drug lead discovery and optimization. *Acc. Chem. Res.* **42**, 724–733 (2009).
2. Verdonk, M. L. & Hartshorn, M. J. Structure-guided fragment screening for lead discovery. *Curr. Opin. Drug Discov. Devel.* **7**, 404 (2004).
3. de Souza Neto, L. R. *et al.* In silico strategies to support fragment-to-lead optimization in drug discovery. *Front. Chem.* **8** (2020).
4. Hoffer, L. *et al.* Integrated strategy for lead optimization based on fragment growing: the diversity-oriented-target-focused-synthesis approach. *J. Med. Chem.* **61**, 5719–5732 (2018).
5. Gerry, C. J. & Schreiber, S. L. Chemical probes and drug leads from advances in synthetic planning and methodology. *Nat. Rev. Drug Discov.* **17**, 333 (2018).
6. Sattarov, B. *et al.* De novo molecular design by combining deep autoencoder recurrent neural networks with generative topographic mapping. *J. Chem. Inf. Model.* **59**, 1182–1196 (2019).
7. Sanchez-Lengeling, B. & Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science* **361**, 360–365 (2018).
8. Jin, W., Barzilay, R. & Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. vol. 80 of *Proceedings of Machine Learning Research*, 2323–2332 (Stockholmsmässan, Stockholm Sweden, 2018).
9. You, J., Liu, B., Ying, Z., Pande, V. & Leskovec, J. Graph convolutional policy network for goal-directed molecular graph generation. In Bengio, S. *et al.* (eds.) *Advances in Neural Information Processing Systems 31*, 6410–6421 (2018).
10. Murray, C. & Rees, D. The rise of fragment-based drug discovery. *Nat. Chem.* **1**, 187–92 (2009).
11. Hajduk, P. J. & Greer, J. A decade of fragment-based drug design: strategic advances and lessons learned. *Nat. Rev. Drug Discov.* **6**, 211–219 (2007).
12. Shi, C. *et al.* Graphaf: a flow-based autoregressive model for molecular graph generation. In *8th International Conference on Learning Representations, Addis Ababa, Ethiopia, April 26-30, 2020* (2020).
13. Zang, C. & Wang, F. Moflow: An invertible flow model for generating molecular graphs. In Gupta, R., Liu, Y., Tang, J. & Prakash, B. A. (eds.) *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, 617–626 (2020).
14. Jin, W., Yang, K., Barzilay, R. & Jaakkola, T. S. Learning multimodal graph-to-graph translation for molecule optimization. In *7th International Conference on Learning Representations, New Orleans, LA, USA, May 6-9, 2019* (2019).
15. Jin, W., Barzilay, R. & Jaakkola, T. S. Hierarchical generation of molecular graphs using structural motifs. In *Proceedings of the 37th International Conference on Machine Learning, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research*, 4839–4848 (2020).
16. Podda, M., Bacciu, D. & Micheli, A. A deep generative model for fragment-based molecule generation. In Chiappa, S. & Calandra, R. (eds.) *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, vol. 108 of *Proceedings of Machine Learning Research*, 2240–2250 (2020).
17. Ji, C., Zheng, Y., Wang, R., Cai, Y. & Wu, H. Graph polish: A novel graph generation paradigm for molecular optimization. *CoRR* **abs/2008.06246** (2020). 2008.06246.
18. Lim, J., Hwang, S.-Y., Moon, S., Kim, S. & Kim, W. Y. Scaffold-based molecular design with a graph generative model. *Chem. Sci.* **11**, 1153–1164 (2020).
19. Ahn, S., Kim, J., Lee, H. & Shin, J. Guiding deep molecular optimization with genetic exploration. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H. (eds.) *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, December 6-12, 2020, virtual* (2020).
20. Nigam, A., Friederich, P., Krenn, M. & Aspuru-Guzik, A. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. In *8th International Conference on Learning Representations, Addis Ababa, Ethiopia, April 26-30, 2020* (2020).
21. Wildman, S. A. & Crippen, G. M. Prediction of physicochemical parameters by atomic contributions. *J. Chem. Inf. Comput. Sci.* **39**, 868–873 (1999).
22. Ertl, P. & Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminf.* **1**, 8 (2009).
23. Sterling, T. & Irwin, J. J. Zinc 15–ligand discovery for everyone. *J. Chem. Inf. Model.* **55**, 2324–2337 (2015).
24. Gómez-Bombarelli, R. *et al.* Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* **4**, 268–276 (2018).
25. Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y. & Martineau, P. An exact graph edit distance algorithm for solving pattern recognition problems. In *Proceedings of the International Conference on Pattern Recognition Applications and Methods - Volume 1*, 271–278 (Setubal, PRT, 2015).
26. Sanfeliu, A. & Fu, K. A distance measure between attributed relational graphs for pattern recognition. *IEEE Trans. Syst. Man Cybern.* **SMC-13**, 353–362 (1983).
27. Lipinski, C. A. Lead-and drug-like compounds: the rule-of-five revolution. *Drug Discov. Today Technol.* **1**, 337–341 (2004).
28. Ghose, A. K., Viswanadhan, V. N. & Wendoloski, J. J. A knowledge-based approach in designing combinatorial or medicinal chemistry libraries for drug discovery. 1. a qualitative and quantitative characterization of known drug databases. *J. Comb. Chem.* **1**, 55–68 (1999).
29. Whiteson, S., Tanner, B., Taylor, M. E. & Stone, P. Protecting against evaluation overfitting in empirical reinforcement learning. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 120–127 (2011).
30. Zhang, C., Vinyals, O., Munos, R. & Bengio, S. A study on overfitting in deep reinforcement learning. *CoRR* **abs/1804.06893** (2018). 1804.06893.
31. Lipinski, C. A., Lombardo, F., Dominy, B. W. & Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Deliv. Rev.* **46**, 3–26 (2001).
32. Rokitskaya, T. I., Luzhkov, V. B., Korshunova, G. A., Tashlitsky, V. N. & Antonenko, Y. N. Effect of methyl and halogen substituents on the transmembrane movement of lipophilic ions. *Phys. Chem. Chem. Phys.* **21**, 23355–23363 (2019).
33. Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S. & Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nat. Chem.* **4**, 90–98 (2012).

34. Olivecrona, M., Blaschke, T., Engkvist, O. & Chen, H. Molecular de-novo design through deep reinforcement learning. *J. Cheminf.* **9** (2017).
35. Kusner, M. J., Paige, B. & Hernández-Lobato, J. M. Grammar variational autoencoder. In Precup, D. & Teh, Y. W. (eds.) *Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6-11 August 2017*, vol. 70 of *Proceedings of Machine Learning Research*, 1945–1954 (2017).
36. De Cao, N. & Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models* (2018).
37. Zhou, Z., Kearnes, S., Li, L., Zare, R. N. & Riley, P. Optimization of molecules via deep reinforcement learning. *Sci. Rep.* **9**, 1–10 (2019).
38. Wainberg, M., Merico, D., Delong, A. & Frey, B. J. Deep learning in biomedicine. *Nat. Biotechnol.* **36**, 829–838 (2018).
39. Kim, S. *et al.* PubChem in 2021: new data content and improved web interfaces. *Nucleic Acids Res.* **49**, D1388–D1395 (2020).
40. Gao, W. & Coley, C. W. The synthesizability of molecules proposed by generative models. *J. Chem. Inf. Model.* (2020).
41. Segler, M. H. S., Preuss, M. & Waller, M. P. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature* **555**, 604–610 (2018).
42. Kishimoto, A., Buesser, B., Chen, B. & Botea, A. Depth-first proof-number search with heuristic edge cost and application to chemical synthesis planning. In Wallach, H. M. *et al.* (eds.) *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, December 8-14, 2019, Vancouver, BC, Canada*, 7224–7234 (2019).
43. Stokes, J. M. *et al.* A deep learning approach to antibiotic discovery. *Cell* **180**, 688–702.e13 (2020).
44. Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
45. Liu, J. & Ning, X. Multi-assay-based compound prioritization via assistance utilization: A machine learning framework. *J. Chem. Inf. Model.* **57**, 484–498 (2017).
46. Liu, J. & Ning, X. Differential compound prioritization via bidirectional selectivity push with power. *J. Chem. Inf. Model.* **57**, 2958–2975 (2017).
47. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 1263–1272 (2017).
48. Xu, K., Hu, W., Leskovec, J. & Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations, New Orleans, LA, USA, May 6-9, 2019* (2019).
49. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. In Bengio, Y. & LeCun, Y. (eds.) *2nd International Conference on Learning Representations, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (2014).
50. Wildman, S. A. & Crippen, G. M. Prediction of physicochemical parameters by atomic contributions. *J. Chem. Inf. Comput. Sci.* **39**, 868–873 (1999).
51. Reddi, S. J., Kale, S. & Kumar, S. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018).
52. Chen, Z. A deep generative model for molecule optimization via one fragment modification. http://doi.org/10.5281/zenodo.4667928 (2021).
53. Krenn, M., Häse, F., Nigam, A., Friederich, P. & Aspuru-Guzik, A. Selfies: a robust representation of semantically constrained graphs with an example application in chemistry. *CoRR* **abs/1905.13741** (2019).
54. Ryu, S., Lim, J., Hong, S. H. & Kim, W. Y. Deeply learning molecular structure-property relationships using attention- and gate-augmented graph convolutional network. *arXiv: Learning* (2018).
55. Hung, A. W. *et al.* Route to three-dimensional fragments using diversity-oriented synthesis. *Proceedings of the National Academy of Sciences* **108**, 6799–6804 (2011).
56. Karypis, G. Cluto a clustering toolkit (2002).
57. Hagberg, A., Swart, P. & S Chult, D. Exploring network structure, dynamics, and function using networkx. Tech. Rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008).
58. Ritchie, T. J. & Macdonald, S. J. The impact of aromatic ring count on compound developability – are too many aromatic rings a liability in drug design? *Drug Discov. Today* **14**, 1011 – 1020 (2009).
59. Lipinski, C. & Hopkins, A. Navigating chemical space for biology and medicine. *Nature* **432**, 855–861 (2004).
60. Gaulton, A. *et al.* The ChEMBL database in 2017. *Nucleic Acids Res.* **45**, D945–D954 (2016).
61. Li, Y., Zhang, L. & Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *J. Cheminf.* **10** (2018).
62. Jin, W., Barzilay, R. & Jaakkola, T. S. Multi-objective molecule generation using interpretable substructures. In *Proceedings of the 37th International Conference on Machine Learning, 13-18 July 2020, Virtual Event*, vol. 119 of *Proceedings of Machine Learning Research*, 4849–4859 (2020).
63. Marler, R. & Arora, J. Survey of multi-objective optimization methods for engineering. *Struct. Multidiscip. Optim.* **26**, 369–395 (2004).
64. Nicolaou, C. A. & Brown, N. Multi-objective optimization methods in drug design. *Drug Discov. Today Technol.* **10**, e427–e435 (2013).
65. Boyd, S., Parikh, N., Chu, E., Peleato, B. & Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **3**, 1–122 (2011).
66. Williams, R. J. & Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Comput.* **1**, 270–280 (1989).
67. Coello, C., Veldhuizen, D. & Lamont, G. *Evolutionary Algorithms for Solving Multi-Objective Problems Second Edition* (2007).
68. Winter, R. *et al.* Efficient multi-objective molecular optimization in a continuous latent space. *Chem. Sci.* **10**, 8016–8024 (2019).
69. Lile, J. R. D., Kang, S. G., Son, Y.-A. & Lee, S. G. Do HOMO–LUMO energy levels and band gaps provide sufficient understanding of dye-sensitizer activity trends for water purification? *ACS Omega* **5**, 15052–15062 (2020).
70. Sivaraman, G. *et al.* A machine learning workflow for molecular analysis: application to melting points. *Mach. learn.: sci. technol.* **1**, 025015 (2020).
71. Sorkun, M. C., Khetan, A. & Er, S. AqSolDB, a curated reference set of aqueous solubility and 2d descriptors for a diverse set of compounds. *Sci. Data* **6** (2019).