nature computational science

Article

https://doi.org/10.1038/s43588-024-00718-0

A deep learning approach for rational ligand generation with toxicity control via reactive building blocks

Received: 9 May 2024

Accepted: 7 October 2024

Published online: 08 November 2024

Check for updates

Pengyong Li[1], Kaihao Zhang[2], Tianxiao Liu[1], Ruiqiang Lu[3], Yangyang Chen[4], Xiaojun Yao[3], Lin Gao[1] & Xiangxiang Zeng[5]

Deep generative models are gaining attention in the field of de novo drug design. However, the rational design of ligand molecules for novel targets remains challenging, particularly in controlling the properties of the generated molecules. Here, inspired by the DNA-encoded compound library technique, we introduce DeepBlock, a deep learning approach for block-based ligand generation tailored to target protein sequences while enabling precise property control. DeepBlock neatly divides the generation process into two steps: building blocks generation and molecule reconstruction, accomplished by a neural network and a rule-based reconstruction algorithm we proposed, respectively. Furthermore, DeepBlock synergizes the optimization algorithm and deep learning to regulate the properties of the generated molecules. Experiments show that DeepBlock outperforms existing methods in generating ligands with affinity, synthetic accessibility and drug likeness. Moreover, when integrated with simulated annealing or Bayesian optimization using toxicity as the optimization objective, DeepBlock successfully generates ligands with low toxicity while preserving affinity with the target.

Searching for ligands, molecules that bind to specific proteins, is a crucial pursuit in drug discovery. Virtual screening has emerged as an important methodology, leveraging computer programs to identify bioactive compounds within small-molecule libraries[1]. However, the efficacy of virtual screening is limited by the vast chemical space and the compound libraries utilized. In contrast, de novo drug design strategies, generating molecular structures from scratch, offer a promising avenue to explore a broader chemical space beyond existing libraries[2,3].

In recent years, deep generative models have made substantial advancements in de novo molecule generation. Various forms of autoregressive[4,5], variational autoencoder[6,7], generative adversarial network[8,9], normalizing flow[10,11] and diffusion model[12,13] have been proposed to generate representations such as simplified molecular input line entry system (SMILES) strings, molecular graphs, or three-dimensional (3D) conformers. These ligand-based drug discovery models aim to learn the underlying distribution of chemical space and subsequently sample novel, valid molecules from this distribution. However, they do not directly generate molecules for specific protein targets, thus necessitating additional screening based on docking or combining with reinforcement learning technology[14].

A recent trend in ligand generation involves considering how molecules interact with protein targets. Methods such as LiGAN[15] and 3D-SBDD[16] integrate structural information derived from protein pockets into molecule generation, tailoring molecules to specific binding targets. However, it is noteworthy that these approaches depend on

[1]School of Computer Science and Technology, Xidian University, Xi'an, China. [2]School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China. [3]Centre for Artificial Intelligence Driven Drug Discovery, Faculty of Applied Sciences, Macao Polytechnic University, Macao, China. [4]University of Tsukuba, Tsukuba, Japan. [5]College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. e-mail: xzeng@hnu.edu.cn

structural information, which may not always be available, particularly for new target proteins with unknown 3D structures.

Furthermore, the synthesizability[17] of generated molecules stands as a paramount concern in practical drug discovery projects, but this aspect is often overlooked in many target-based ligand generative models. Although generated molecules may theoretically show strong affinity, their practical synthesis could pose substantial challenges, thereby limiting the applicability of the generative model. In addition, most existing methods concentrate solely on the affinity between the generated molecules and the target, neglecting factors such as toxicity, metabolism and other essential attributes crucial for the successful market approval of a drug.

The DNA-encoded compound library[18] technique has emerged as a widely accepted wet-lab method for hit finding. This method uses combinatorial chemistry[19] to facilitate the rapid generation of numerous candidate compounds through the reaction of molecular building blocks. Inspired by the DNA-encoded compound library technology, we introduce a deep learning-based framework named DeepBlock for de novo drug design, utilizing molecular building blocks. Here building blocks denote molecular fragments capable of chemically reacting with one other. The core concept of DeepBlock involves decomposing the molecule-generation process into two sequential steps: first, generating the building blocks based on the protein sequence embedded features, and subsequently assembling them into complete molecules. By harnessing the intrinsic properties of these blocks and the chemical interactions among them, DeepBlock can design rational molecules of superior quality. Building on this concept, we have devised effective mechanisms in DeepBlock to address two key tasks: molecule generation tailored to the protein sequence and property control during the generation process.

In DeepBlock, we introduce the Block Generative Network (BGNet), a conditional deep generative model designed to generate the sequence of blocks based on a given protein sequence. BGNet incorporates two key features that substantially enhance its performance. First, it is constructed by a molecular block autoencoder pretrained on a large-scale molecule dataset. This pretraining expands the chemical space and alleviates the potential overfitting stemming from the limited size of the protein–ligand pairs dataset. Second, we introduce a critical component, the target contribution perception module, into DeepBlock. This module enhances the model's ability to autonomously identify interactions between ligands and residues, compensating for the absence of 3D structural information within protein sequences. The combination of these two features within BGNet underscores its capacity in generating diverse and bioactive molecular fragments, effectively addressing the challenges posed by protein sequence data. Moreover, we use BGNet in conjunction with a simulated annealing (SA) algorithm[20] or Bayesian optimization (BO) to control the generation process, aiming to enhance additional properties while preserving their binding affinity for the target protein. We conduct experiments focusing on drug toxicity as the optimization objective and demonstrate that our framework can generate molecules with reduced toxicity levels.

Results

The DeepBlock framework

In this study, we propose a deep learning-based framework, named DeepBlock, to generate and optimize the ligand molecules (Fig. 1a) by decomposing the generation process into two steps: the generation of building blocks and the molecule reconstruction. We first design a molecular fragmentation and reconstruction algorithm (Fig. 1e), which aims to convert ligand molecules in the dataset into blocks sequences for training the block generator BGNet (Fig. 1b), and reconstruct the generated blocks back into valid molecules. To enrich the model's molecular understanding, BGNet uses a dual encoding scheme for ligands and proteins, coupled with a binding contribution perception network that predicts the importance of each protein residue in ligand binding. This approach ensures that residues of binding sites are deemed more important for the interaction, directly influencing the target representation. Enhancements in DeepBlock's predictive capabilities are achieved through self-supervised pretraining on a large-scale molecule dataset and by incorporating evolutionary scale modeling (ESM-2)[21], a protein language model, for in-depth feature extraction from protein sequences (Fig. 1a). When generating, BGNet takes a protein sequence as input condition T and generates the ligand's blocks sequence variable B (Fig. 1c). Consequently, we integrate BGNet with the optimization algorithms, leveraging BGNet for generating neighboring candidates and using the SA or BO algorithm for exploration, thereby enabling controlled manipulation of molecular properties during the generation process (Fig. 1d).

Molecule fragmentation and reconstruction

The Breaking of Retrosynthetically Interesting Chemical Substructures (BRICS) algorithm[22] defines a series of cleavage bonds based on retrosynthetic chemistry and is widely used as guiding rules in molecular fragmentation. However, the combinatorial explosion inherent in graph structures renders the reconstruction of the initial molecule unattainable solely through the blocks cleaved by BRICS. Here we propose a graph-based fragmentation and reconstruction algorithm (Fig. 1e) to implement the transformation process of 'molecule → block sequence → molecule', with the stringent constraint that the molecule remains identical at the beginning and at the end. More details can be found in 'Molecule fragmentation and reconstruction algorithm' in Methods. To validate the reliability, we applied this conversion for each molecules in ChEMBL dataset[23]. Out of 2,205,345 molecules, only 70 molecules failed to convert (further analysis is provided in Supplementary Figs. 1–5), resulting in a success rate of approximately 99.99683%. This high success rate demonstrates the reliability and practicality of our approach, which effectively supports downstream model learning tasks.

Molecular generation

Evaluation and comparison. As shown in Fig. 2a, the molecules generated by DeepBlock show comparable docking affinity to TargetDiff and Pocket2Mol, and outperform those generated by the other baseline models. In addition, the distribution of the Vina scores (Fig. 2d) reveals that Pocket2Mol and TargetDiff show a larger variance with more outliers, whereas DeepBlock shows a more concentrated distribution with fewer outliers. This indicates that DeepBlock generates more consistent and reliable candidates. It is worth noting that a molecule with a high docking score may contain uncommon or infeasible substructures, which are unacceptable in drug development. Therefore, we further analyzed the quantitative estimate of drug-likeness (QED) and Retro* scores of molecules with high docking affinity (Vina score <−7.445) and visualized the results in a scatter plot in Fig. 2c. Pocket2Mol tends to generate high-affinity molecules with poor QED or synthetic accessibility, and TargetDiff's high-affinity molecules also suffer from poor synthetic accessibility. This indicates the potential presence of unrealistic structures in the molecules generated by these methods. In contrast, our method yields molecules that not only achieve high docking affinity but also show superior drug-like properties and synthetic feasibility. Furthermore, as shown in Fig. 2a, DeepBlock substantially outperforms the baselines in terms of the prediction success rate of Retro*, which ensures the practicality of subsequent wet-lab validation. Moreover, the molecules generated by DeepBlock show the highest fragment similarity to known reference ligands but lower scaffold similarity. In addition, as shown in the distribution of molecular physicochemical properties in Fig. 2b, the distribution of molecules produced by DeepBlock closely resembles that of known ligands, suggesting a strong alignment with established molecular features. In terms of diversity, our model performs at an intermediate level. However, in terms of inference time, we can generate candidate

https://doi.org/10.1038/s43588-024-00718-0

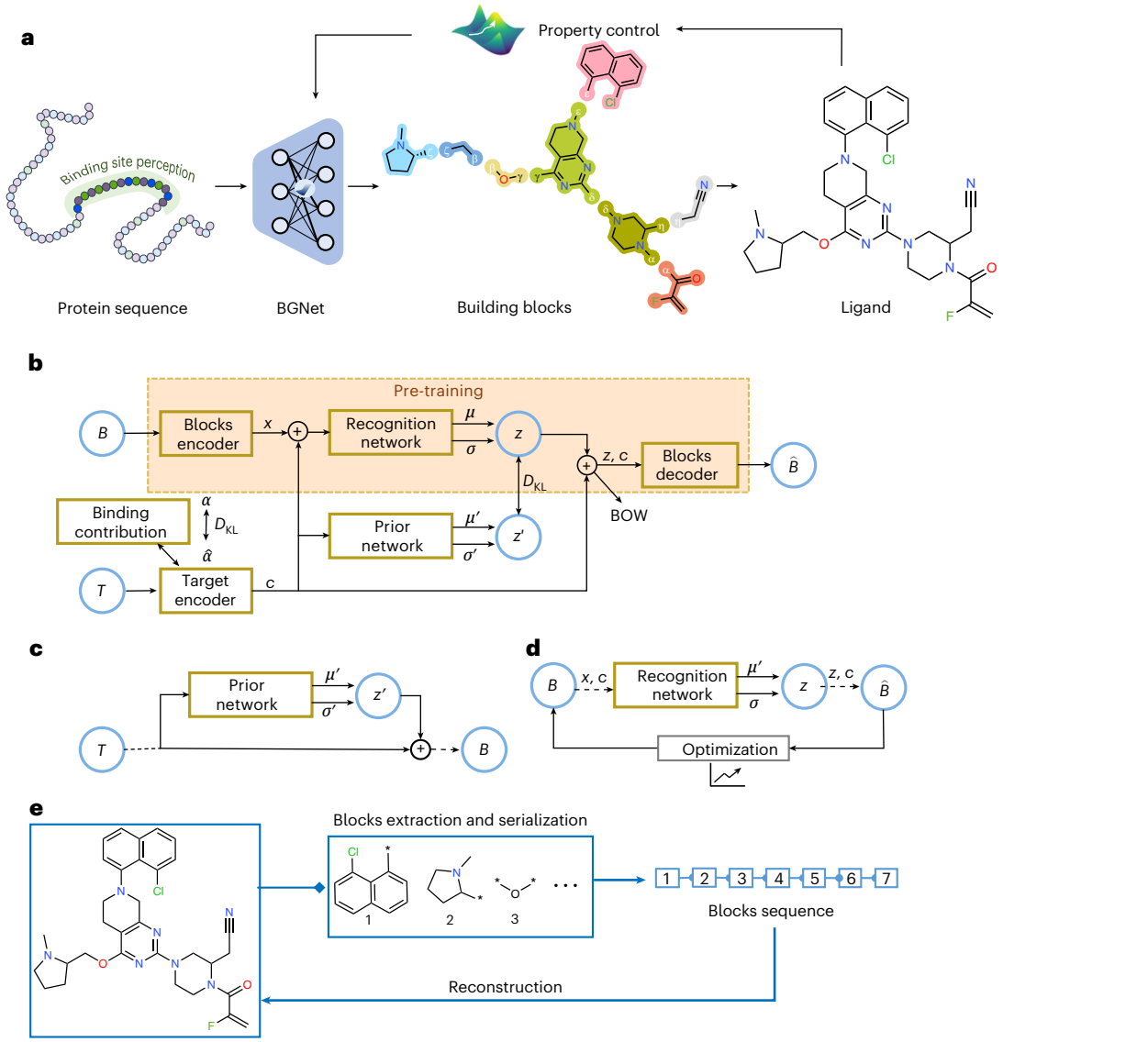


Fig. 1 | Overview of the DeepBlock framework. a, The workflow of DeepBlock. Different colors are used for the building blocks to distinguish the blocks. Greek letters are used to denote the dummy atoms, also referred to as breakpoints, as they represent where the bonds were broken. **b**, The overall architecture of DeepBlock. x and c represent the learned representation of the blocks sequence variable B and target protein variable T, respectively. z and z' are the latent vectors of B and T sampled from distributions with means μ and μ', and variances σ^2 and σ'^2. $\hat{\alpha}$ and α are the predicted contribution score and the ground-truth score. D_{KL} represents the Kullback–Leibler divergence. BOW represents bag-of-words loss. The areas highlighted with an orange background represent the pretrained components of the model. The light blue circles represent the input and output vectors and the dark yellow boxes indicate the neural network. **c**, The generation process of DeepBlock. **d**, The property control during the generation. Optimization represents the optimization algorithm, including SA and BO. **e**, Molecular fragmentation and reconstruction. Numerals 1 to 7 denote different building blocks.

molecules faster compared with the baseline models, ensuring a balance between quality and efficiency. As illustrated in Fig. 2e, the pretraining scheme used in DeepBlock enhanced the validity, novelty and uniqueness of the generated molecules. In addition, we evaluated DeepBlock on an alternative dataset, PDBbind[24]. The results (Supplementary Table 1) showed that DeepBlock continues to perform well, indicating its robustness and generalizability across different datasets.

Binding contribution. In DeepBlock, we defined a residue's binding contribution coefficient, which is negatively correlated with the distance between the residue and the protein pocket center, and developed a neural network to automatically predict it (see 'Binding contribution of the residue' in Methods for more details). Here we analyzed the correlation between the predicted contribution values of the model with the ground truth. The calculated Pearson correlation coefficient, amounting to 0.68 with a statistically significant P value <0.001, denotes a positive correlation between the predicted and actual values. Illustrating with the ABL2 protein as an example, we depict the comparative distribution of predicted values against the actual values in Fig. 3a (Supplementary Figs. 6–11 for other test proteins). The close alignment of the two curves therein underscores the model's aptitude in accurately capturing the relative magnitudes of contribution coefficients. As shown in Fig. 3b, the overall estimated contribution values of the binding site are higher than that of other protein residues. However, for some individual proteins and residues, their estimated contribution values do not strictly follow the distance-based negative correlation rule. This phenomenon is due to two main factors. First, the inherent margin of error in contribution value prediction, an auxiliary task that the model may not fully learn. Second, the model might detect multiple binding sites, whereas the ground truth reflects

https://doi.org/10.1038/s43588-024-00718-0

a

	Reference	LiGAN	3D-SBDD	Pocket2Mol	TargetDiff	DeepBlock
Vina score (↓)	−7.445 ± 2.276	−6.142 ± 0.847	−6.928 ± 1.558	−7.221 ± 2.132	**−7.284 ± 1.418**	−7.224 ± 0.971
QED (↑)	0.476 ± 0.206	0.460 ± 0.069	0.508 ± 0.120	**0.585 ± 0.098**	0.483 ± 0.108	0.543 ± 0.077
SA (↓)	3.453 ± 1.259	4.798 ± 0.278	4.304 ± 0.857	3.194 ± 1.172	4.546 ± 0.143	**3.186 ± 0.357**
Retro* (%, ↑)	58.00 ± 49.60	7.69 ± 5.13	21.88 ± 19.32	54.29 ± 33.43	22.02 ± 22.99	**60.27 ± 13.90**
Diversity (↑)	–	**0.834 ± 0.031**	0.697 ± 0.115	0.727 ± 0.161	0.742 ± 0.093	0.784 ± 0.071
log*P* (–)	0.894 ± 2.742	0.595 ± 0.766	0.406 ± 1.591	2.029 ± 1.477	1.768 ± 1.378	**1.112 ± 1.244**
Fragment similarity (↑)	–	0.109 ± 0.124	0.176 ± 0.221	0.116 ± 0.144	0.173 ± 0.177	**0.205 ± 0.231**
Scaffold similarity (↑)	–	0.152 ± 0.101	**0.221 ± 0.142**	0.154 ± 0.140	0.147 ± 0.131	0.144 ± 0.217
Time (s, ↓)	–	1,741	3,139	1,962	2,558	9

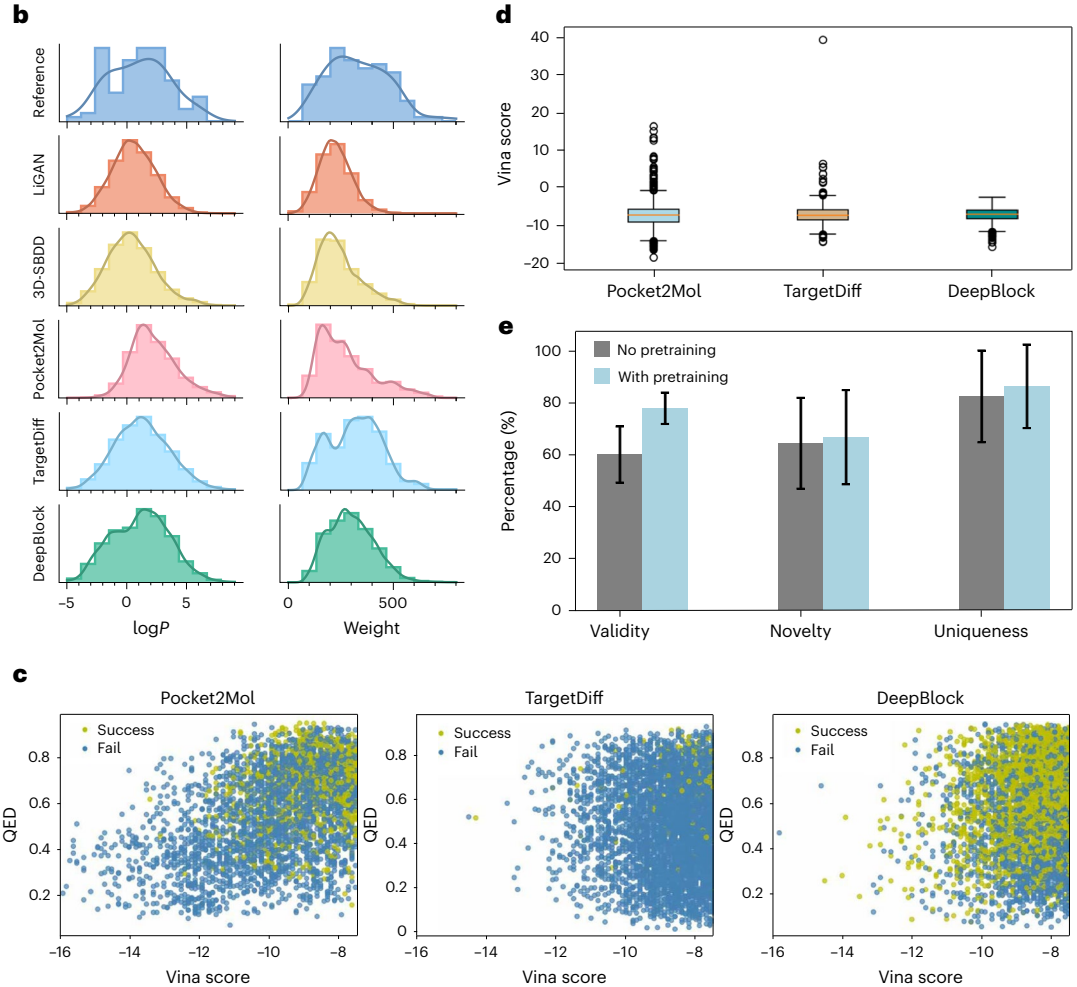
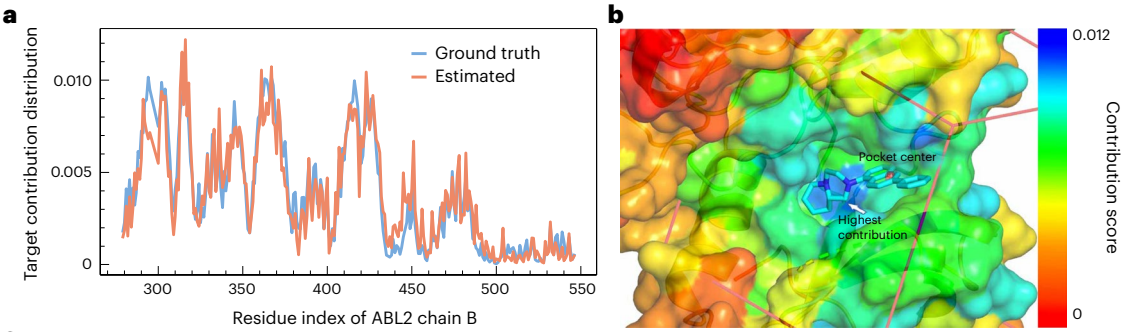


Fig. 2 | Comparison of the proposed method and baselines. a, Comparison metrics of the reference molecules, as well as the molecules generated by the baseline and our method, where bold indicates the best value among our results and the baselines. We first calculate the mean within each group of samples from each protein separately, and then calculate the overall mean ± s.d. (unbiased). Bold formatting denotes the optimal values. **b**, The distribution of log*P* and molecular weight for all generated molecules and reference molecules. **c**, Scatter plot of generated molecules. Each spot represents a molecule, colored by its Retro* score. Red represents molecules that failed to predict a synthetic route via Retro*, indicating they are difficult to synthesize. Green represents molecules that successfully predicted a synthetic route via Retro*, indicating they are easy to synthesize. **d**, Box plot of Vina scores for generated molecules. Center lines denote median values; whiskers denote 1.5 times the interquartile range; the upper and lower limits of the box plot indicate the maximum and minimum values, respectively. **e**, Comparison of performance between with and without pretraining. All values are shown as mean ± s.d. (N = 20,000 molecules).

only one, leading to discrepancies. In the absence of known target structures, we also assess the quality of generated molecules based on automatically predicted contribution scores, as opposed to real values derived from structural information. As shown in Fig. 3c, the affinity of molecules generated with predicted contribution scores was slightly lower than that of molecules generated using structural information, and showed comparable drug likeness, synthetic accessibility and similarity to known ligands. This subtle difference in affinity further validates the accuracy and reliability of our contribution-prediction module.

In addition, we conducted ablation experiments on the contribution-prediction module. As shown in Fig. 3c, the docking

https://doi.org/10.1038/s43588-024-00718-0



c

	Vina score (↓)	QED (↑)	SA (↓)	Retro* (↑)	Diversity (↑)	Fragment similarity (↑)	Scaffold similarity (↑)
DeepBlock (predict)	−7.164 ± 0.974	0.539 ± 0.081	3.185 ± 0.394	60.29 ± 14.32	0.788 ± 0.075	0.190 ± 0.217	0.213 ± 0.160
DeepBlock (without)	−7.184 ± 0.942	0.539 ± 0.084	3.201 ± 0.376	60.42 ± 13.34	0.784 ± 0.079	0.194 ± 0.220	0.142 ± 0.213
DeepBlock	−7.224 ± 0.971	0.543 ± 0.077	3.186 ± 0.357	60.27 ± 13.90	0.784 ± 0.071	0.205 ± 0.231	0.144 ± 0.217

Fig. 3 | Binding contribution of a residue. a, Comparison of the ground truth and estimated distributions of target contributions, using ABL2 (Protein Data Bank ID 4XLI) as an example. **b**, Visualization of the target contribution. The color bar represents the contribution score, with blue indicating higher scores, red indicating lower scores and yellow representing intermediate. The scores range from 0.0 (lowest) to 0.012 (highest). The red wireframe represents the simulated docking pocket, with dimensions of 25 Å in length, width and height. The target center region shows higher predicted contribution values. In the 3D molecular structure, C atoms are represented in cyan and N atoms in blue. **c**, Performance comparison of DeepBlock with predicted contribution coefficient (DeepBlock (predict)), DeepBlock without contribution-prediction module (DeepBlock (without)) and DeepBlock. (↑) indicates higher is better. (↓) indicates lower is better.

affinity of DeepBlock without the contribution-prediction module is slightly lower than that of DeepBlock with the module. In addition, the drug likeness, synthetic accessibility and similarity to known ligands also showed a minor decline. The difference in docking affinity is not substantial, which can be attributed to the fact that the contribution-prediction module does not introduce additional information but rather aids in learning the representation of protein sequences. Therefore, although the contribution-prediction module can enhance the quality of generated molecules, one of its important roles is serving as an auxiliary tool for analyzing protein binding sites.

Case study of ligand generation for a novel target. To demonstrate DeepBlock's capacity for ligand generation targeting a novel protein without structural information, we chose the KIAA1363 target as a case study for analysis. KIAA1363 (gene name *NCEH1*) is a serine hydrolase and has been observed to show a positive correlation with the invasiveness of tumor cells[25]. The identification of active inhibitors targeting KIAA1363 presents a promising therapy approach toward cancer[26,27]. Here we use DeepBlock to generate 100 ligands based on the amino acid sequence of KIAA1363, and subsequently select 5 molecules with the highest docking affinity. As shown in Fig. 4a, the generated five sample molecules have similar substructures and docking pockets to the known inhibitor JW480[27], accompanied by high docking affinity, favorable drug-like properties and synthesizability, indicating the potential of DeepBlock in target sequence-based ligand generation.

Furthermore, we conducted long-term dynamics simulations to verify the binding stability. From the ligand's root mean square deviation (RMSD) curves (Fig. 4c), both JW480 and the generated molecules showed stable binding to the ligand. Further analysis of the trajectories revealed that JW480 primarily formed unstable van der Waals interactions with Gly114 and Leu36 of KIAA1363. In contrast, our generated molecules formed more stable van der Waals interactions with Gly114, and hydrogen-bond interactions were also observed. The generated molecules were able to form key interactions in the binding pocket that are similar to, but more stable than, those of the known inhibitors. In addition, we selected a well-characterized target, severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) main protease (Mpro), for validation. Molecular dynamics simulations further demonstrated the binding stability of the molecules generated by DeepBlock with the Mpro target (Supplementary Fig. 12).

To investigate the relationship between the block reconstruction mechanism of the model and the retrosynthetic pathway, we visualized the process of reconstructing the predicted block sequence of the model along with the retrosynthetic pathway predicted by Retro* (top-left sample in Fig. 4a). On the left side of Fig. 4d, the blocks decoder in DeepBlock generates a block sequence of length 6, labeled as **1** to **6**. On the right side of Fig. 4d, Retro* predicts a two-step reaction pathway for three reactants. It is worth noting that the three reactants designed in the reaction can directly correspond to certain blocks in the active blocks. This indicates that the chemical bond cleavage method used in the earlier stages can accurately act on the reactive chemical bonds while preserving the chemical significance of the substructures. Furthermore, the corresponding blocks of the reactants also maintain a contiguous relationship in the original block sequence, suggesting that the model is capable of effectively combining blocks and expressing structural information.

Molecular property control

Binding affinity optimization. Here we selected 'F16P1 (3kc1)' from the test set of CrossDocked 2020 (ref. 28) as the target receptor and randomly chose 5,000 small molecules from the ChEMBL[23] dataset as the initial molecules for optimization (refer to 'Binding affinity optimization' in Methods for details). Each data point in Fig. 5a represents the optimization results of molecules within a specific binding affinity range (pre-optimization affinity ±0.5). For example, among the molecules with pre-optimization affinity in the range of −7 ± 0.5, 57.53% of the molecules were successfully optimized, and the average similarity value before and after optimization was 0.307. It can be observed that molecules with lower initial binding affinity better reflect the optimization effect of the model. Although these molecules underwent modifications to improve the docking ability, the changes in the similarity curve are still relatively smooth, ensuring a certain level of molecular similarity. In Fig. 5b, we provide a more intuitive demonstration of the numerical changes in low-affinity molecules after optimization. It can be seen that the model effectively optimizes molecules with initial affinities greater than −7.2, maintaining docking scores of around −7.

https://doi.org/10.1038/s43588-024-00718-0

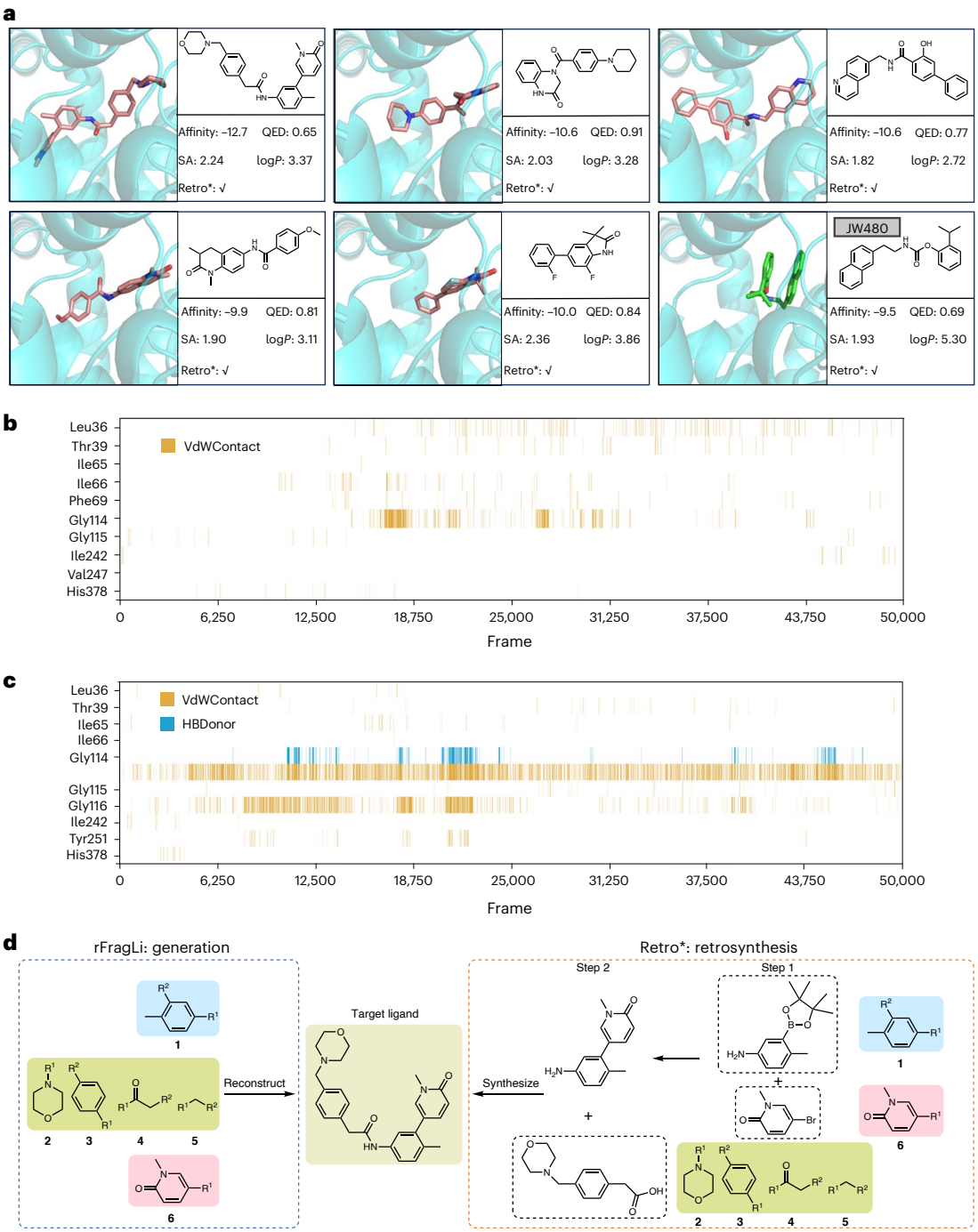
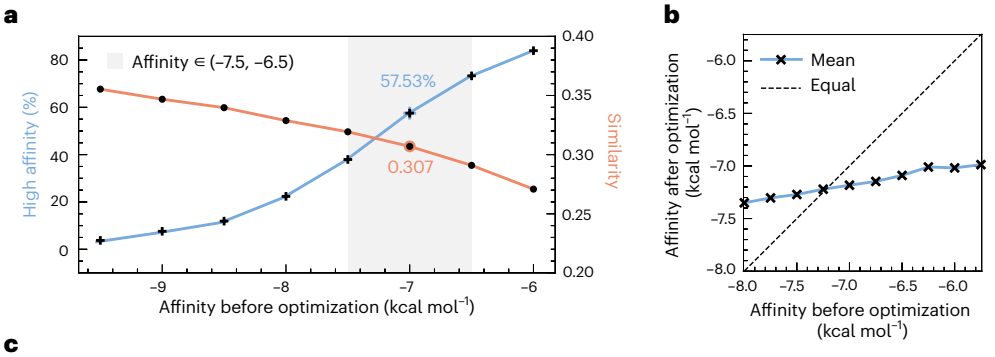


Fig. 4 | Case study of ligands generation for KIAA1363. a, Generated ligand samples for the target KIAA1363, showing their 3D conformation after docking (C atoms are pink, N atoms are blue and O atoms are red). Compared with inhibitor JW480 (C atoms are green, N atoms are blue and O atoms are red). We also calculated affinity and other metrics for them, where a tick mark for Retro* indicates a successfully predicted retrosynthetic pathway. **b**, Residue-level protein–ligand interaction of JW480. **c**, Residue-level protein–ligand interaction of the DeepBlock-generated molecule with the highest affinity (the molecule in the top left of **a**). VdWContact and HBDonor denote van der Waals contact and hydrogen-bond donor, respectively. **d**, Synthetic accessibility of the generated ligand. On the left is the molecular block we generated, in the middle is the reconstructed molecule and on the right is the retrosynthetic pathway predicted by Retro*.

Using the median value of pre-optimization binding affinity, −7.2, as a threshold, we further analyzed the changes in molecular properties before and after optimization for 2,346 sets of low-affinity molecules with pre-optimization affinity greater than −7.2. As shown in Fig. 5c, 76.04% showed an improvement in binding affinity, with an average difference of more than 0.5 kcal mol^{-1}. Although the drug likeness and synthesizability of the predicted results slightly decreased compared with the original 5,000 small molecules, which are already drug-like compounds, the average values of Lipinski's rule-of-five satisfaction increased slightly, indicating a high potential for drug likeness. For this subset of molecules, the average similarity before and after optimization was 0.288 ± 0.125. As a reference, we also tested the average similarity of randomly selected pairs of 5,000 molecules from ChEMBL, which was 0.268 ± 0.086. Considering the modifications made to

https://doi.org/10.1038/s43588-024-00718-0



c

	Before optimization	After optimization	Better properties (%)
Affinity (kcal mol^{-1}, ↓)	−6.515 ± 0.518	−7.086 ± 0.827	76.04
QED (↑)	0.620 ± 0.227	0.589 ± 0.209	45.91
SA (↓)	2.602 ± 0.664	2.769 ± 0.773	45.27
log*P* (–)	3.051 ± 1.896	2.505 ± 2.049	–
Lipinski (↑)	4.586 ± 0.907	4.662 ± 0.722	85.81

Fig. 5 | Comparison of affinity before and after optimization. a, Each sample point represents the optimization of a molecule within a specific affinity range (±0.5 before optimization). Molecules with lower initial affinity can better reflect the optimization effect of the model. **b**, The numerical changes after optimization of low-affinity molecules more intuitively. **c**, Optimization performance on molecules with low binding affinity. The docking ability of the majority of the molecules was improved.

low-affinity molecules, the preservation of structural features by the model for medium to low-affinity molecules remains within the desired range. Finally, we randomly selected a subset of optimized molecules for visualization (Supplementary Fig. 13). These molecules showed varying degrees of improvement in binding affinity, but the fluctuations in other properties were not consistent. It is worth noting that the structures of the molecules before and after optimization are highly similar, with key substructures preserved.

Target-aware molecular optimization. We propose two target-aware molecular optimization methods—SA-based (SATMO) and BO-based (BOTMO)—which optimize in discrete molecular space and molecular latent space, respectively. Unlike existing molecular optimization approaches, our methods introduce target-aware molecular optimization that incorporates target constraints, ensuring that the generated molecules maintain affinity with the target throughout the optimization process. Figure 6 shows that both optimization methods effectively improve toxicity and QED, while preserving or even slightly enhancing docking affinity and synthetic accessibility. The performance of the BOTMO method is influenced by the number of new candidates N generated during the iterative process. As shown in Fig. 6a, a larger N generally leads to better optimization results. However, increasing N also prolongs the optimization time (Fig. 6d). When $N = 100$, the optimization results are comparable to those of SATMO, but the optimization speed is slower. Furthermore, in Fig. 6g, we present the distribution of molecular energy and various properties before and after optimization for SATMO. It can be observed that there is a noticeable decrease in energy and toxicity after optimization. However, the affinity shows two peaks in the distribution after optimization, possibly due to the trade-off between toxicity and affinity. Regarding drug likeness, compared with before optimization, the distribution after optimization shows a peak above 0.5, with a higher proportion of highly drug-like molecules on the right side.

Discussion

DeepBlock harnesses the power of reactive building blocks to facilitate the rational design of molecules with high drug likeness, synthesizability and docking affinities. The quality of the reactive blocks is a pivotal factor in determining the overall quality of the reconstructed molecule. In our approach, we constructed an extensive dictionary containing 10,701 blocks, which encompasses a wide range of frequently used fragments. Complemented by pretraining procedures, we substantially expanded the chemical space within our model. This process mitigates the risk of overfitting inherent to the limited size of the protein–ligand pairs dataset, consequently enhancing the overall performance of the model. However, it is essential to acknowledge the limitation of our method. DeepBlock is currently capable of generating blocks solely from the existing block dictionary, thereby limiting the diversity of the molecules it generates. A future avenue of our research involves exploring methods for the de novo generation of blocks, thereby liberating our model from the confines of existing block dictionaries and unlocking the potential for greater versatility and novelty in the molecules it can create. Furthermore, DeepBlock generates two-dimensional (2D) molecular structures as SMILES strings, offering controllable properties and applicability to new targets. While SMILES strings provide sufficient structural information for various drug development scenarios, they lack 3D structural details. Future research will focus on integrating our method with approaches such as LiGAN to develop controlled 3D molecular generation methods based on molecular building blocks. This hybrid approach could combine the strengths of both 2D and 3D drug design methodologies, potentially enhancing drug discovery efficiency and effectiveness.

Methods

Dataset

Protein–ligand pairs data for training. To train the entire model, we used the CrossDocked 2020 dataset[28], which initially contained 22.5 million protein–ligand pairs. Following the same filter and split method as described in ref. 16, we filtered out data points with a binding pose RMSD greater than 1 Å, resulting in a refined subset of 184,057 data points. To ensure no overlap between training and validation sets, we used mmseqs2[29] to cluster the data at 30% sequence identity. From these clusters, we randomly selected 100,000 protein–ligand pairs for training and 100 proteins from the remaining clusters for testing. Furthermore, we used 20% of the training data for hyperparameter tuning and model selection. The final results reported in our paper

https://doi.org/10.1038/s43588-024-00718-0

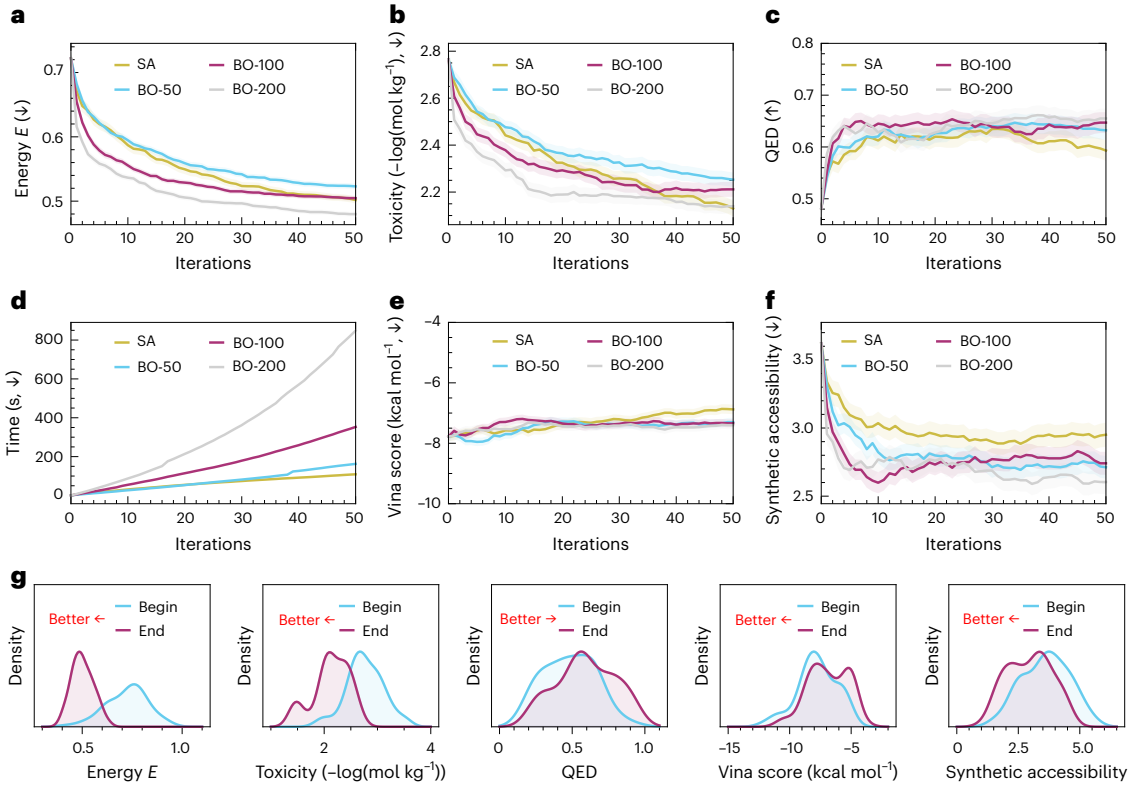


Fig. 6 | The optimization processes and results. a–**c**, The changes in the energy function (**a**), toxicity (**b**) and QED (**c**) during the optimization iterations, collected from 100 independent optimization processes. BO-50, BO-100, and BO-200 refer to BO-based methods that generate $N = 50$, 100, and 200 new candidates, respectively, during the iterative process. The widths of the shadings represent the standard deviation from the mean ($N = 100$ independent optimization processes, with 0.1× scaling applied to the standard deviation in the figures for better visualization). **d**, The time elapsed from the start of the experiment to the end of each iteration. **e**,**f**, The Vina score (**e**) and synthetic accessibility (**f**) of the molecules at each iteration. Mean model scores with standard deviation are shown ($N = 100$ independent optimization processes). **g**, The distribution of molecular properties before and after optimization by the SA algorithm.

are based on the performance on the test set, ensuring that there is no data overlap between the training and testing phases. After splitting, we extracted the protein sequences and molecular SMILES strings. In addition, we utilized structural information to calculate amino acid residue contribution scores for the training of binding contribution-prediction module (refer to 'Binding contribution of the residue').

Drug-like molecule data for pretraining. In pretraining, we use a large-scale dataset of drug-like molecules. ChEMBL is an open database maintained by the European Bioinformatics Institute for drug discovery[23]. It contains a vast amount of drug and bioactivity data, including chemical structures, physicochemical properties and pharmacological activity data. The dataset version we used is CHEMBL31, which includes approximately 2.3 million drug molecules represented as SMILES strings.

Tokenization and embedding. *Molecule or ligand*. We processed each drug-like molecule or ligand molecule as described in 'Molecule fragmentation and reconstruction algorithm' to obtain block sequences. For ChEMBL, we kept only blocks that appeared more than 100 times (Supplementary Fig. 14), resulting in 5,109 unique blocks. For CrossDocked, we retained all blocks, amounting to 7,483. After filtering, deduplication, merging and sorting (Supplementary Table 2), we obtained a block dictionary with a vocabulary size of 10,701 (including 4 special words). For the detailed processing procedure, refer to Supplementary Fig. 14 and Supplementary Table 2. We randomly initialized the embedding layer parameters of the ligand encoder or decoder model with this dimension size.

Protein. Protein sequences are typically encoded using the 20 types of amino acid, making it easy to construct a dictionary. We directly utilized the built-in transformation tools of ESM-2 to achieve end-to-end embedding vector (amino acid features) generation.

Molecule fragmentation and reconstruction algorithm

We have developed a reliable method for converting molecules to block sequences that is reversible and orthogonal to various bond-breaking rules. (1) By utilizing the BRICS rules[22], we track breakable bonds and extract all indivisible minimizable blocks in a single step. These blocks are then abstracted into chemically independent graph structures. Next, we design a graph search method to convert the graph into a sequence. The order of node traversal is strictly defined by the blocks' relative mass and the breakpoints' atom index. (2) During the process of reconstructing the molecule, we sequentially read the blocks and identifiers from the sequence to construct the graph. We extract the broken chemical bonds and connect the blocks to form a complete molecule.

To ensure uniqueness in our decomposition approach, we use the following methods. (1) BRICS rules: we apply BRICS rules to identify breakable bonds and sever them, generating the smallest indivisible substructures or blocks. This ensures a consistent and repeatable decomposition across different molecules. (2) Breakpoint ordering: during the serialization of blocks into sequences, we sort breakpoints in ascending order based on the RDKit atom index, derived from the order of atoms in the SMILES arbitrary target specification (SMARTS) expressions. (3) Graph traversal starting point: we initiate graph traversal from the block with the highest relative mass, with the order of traversal determined by the previously established breakpoint ordering.

These steps collectively ensure a unique decomposition for each molecule. In the following section, we elaborate on the methodologies for molecular fragmentation and reconstruction in detail.

From molecule to block sequence. For illustrative purposes, we demonstrate the fragmentation method using the example of the $KRAS^{G12C}$ inhibitor drug molecule, Adagrasib[30], as shown in Supplementary Fig. 15. In Supplementary Fig. 15b, the BRICS rules indicate the presence of seven breakable chemical bonds in Adagrasib, denoted by lowercase letters. We have also highlighted the atoms at both ends of the breakable bonds, with the numbers in the figure representing the index of each atom.

By breaking all seven chemical bonds and connecting a dummy atom to each end of the bond, we obtain eight blocks, as shown in Supplementary Fig. 15c. These blocks are referred to as nodes and are represented by numbers N. The symbol for the dummy atom n is represented by a Greek letter (for example, α, β and so on), and in this context, we refer to these dummy atoms as breakpoints. In summary, a node contains multiple inequivalent breakpoints, and these breakpoints are necessarily connected to a breakpoint in another node. This results in a rather unique graph structure in Supplementary Fig. 15d.

For each node, to differentiate between different breakpoints within it, we sort the breakpoints in ascending order based on the RDKit[31] atom index, which is determined by the order of atoms in the SMARTS expressions. Each breakpoint within a node is re-assigned a breakpoint label x sequentially, where $x = \alpha, \beta, \ldots, x_{P_N}$, and P_N represents the total number of breakpoints in block N.

For the SMARTS expression of the blocks (nodes), the encoding of a dummy atom (breakpoint) is converted to a valid SMARTS notation as 'a number*' (for example, α is replaced with '1*' and β is replaced with '2*'). In addition, we aim to ensure the uniqueness of block representations, meaning that blocks with the same chemical structure should have consistent SMARTS expressions when the encoding is removed.

For each broken chemical bond, we construct a quadruplet $[N_1, x, N_2, y]$, indicating that breakpoint x on node N_1 is connected to breakpoint y on node N_2. If a node M has an edge with the current node N, we refer to M as an adjacent node of N.

In this way, we can obtain a set of quadruplets from the broken molecule, consisting of 7 edges: {[1, α, 2, α], [2, β, 3, α], [2, γ, 8, α], [3, β, 4, β], [3, γ, 7, α], [4, α, 5, α], [5, β, 6, α]}. We have also determined the number of breakpoints contained in each of the eight nodes. We further utilize a graph search method to serialize them into a unique and reversible node sequence.

The pseudocode for the graph search method is shown in Supplementary Algorithm 1 and Supplementary Fig. 15e. (1) Convert the set of quadruplets $\mathbf{Q} = \{[N_i, x, N_j, y], [N_j, y, N_k, z], \ldots\}$ into a mapping f_Q that can query the adjacent node connected to breakpoint x of any node N_i. Choose the block with the highest relative mass as the root node N_R to start traversing the graph. At this point, the parent node N_F of N is empty, meaning $N = N_R$ and $N_F = \varnothing$. (2) Add the node N to the sequence $\mathbf{L}$. Traverse the breakpoints of node N in the order of breakpoint numbers $x = \alpha, \beta, \ldots, x_{P_N}$. Use the mapping f_Q to find the adjacent node N_C of N. (3) If N_C is the parent node N_F of node N, add the identifier '_' to the sequence $\mathbf{L}$. Otherwise, set $N = N_C$, $N_F = N$, and continue recursive search.

Taking node 2 as an example, it contains 3 breakpoints represented by the blue strings 2α, 2β and 2γ. The notation 1α → 2α indicates that the algorithm, through the mapping f_Q, queries that breakpoint α of node 1 is connected to breakpoint α of the adjacent node 2. During the recursive process, the algorithm continuously discovers child nodes according to the order of breakpoints and records them sequentially. In the end, we obtain a sequence $\mathbf{L}$ = (7 → 3 → 2 → 1 → _ → _ → 8 → _ → 4 → 5 → _ → 6 → _ → _ → _) of length 15.

From fragment sequence to molecule. The method for reconstructing the molecule from the sequence by building the graph is represented as Supplementary Algorithm 2. (1) Set the first node of the sequence $\mathbf{L}$ as the root node. At this point, initialize $i = 1, N = \mathbf{L}_1, N_F = \varnothing, y = \varnothing$. (2) Traverse the breakpoints of N and increment the pointer i. Retrieve the adjacent node $N_C \leftarrow \mathbf{L}_i$. If N_C is a identifier symbol '_', add the quadruplet $[N_F, x, N, y]$ to the set. Otherwise, set $N = N_C$, $N_F = N$, $y = x$, and recursively traverse the adjacent node N_C. Finally, based on the connectivity information contained in the quadruplets, we remove the dummy atoms, connect the valid atoms on both sides and obtain the complete molecule.

Shorten the block sequence. In the example of Adagrasib in this paper, we obtained a sequence of length of 15, but it contains 7 identifiers. From Supplementary Fig. 15d, we can identify nodes 1, 8 and 6 as graph-theoretical leaf nodes, which only have a identifier connected to the parent node at breakpoint α. In the absence of this symbol, it is still possible to determine that breakpoint α is the parent node's connecting breakpoint. Moreover, shorter sequences are more conducive to model learning and reduce computational costs.

On the basis of these considerations, we generalize the condition for omitting identifiers as condition 1: the breakpoint connected to the parent node is the last breakpoint of the current node. Furthermore, condition 2: if the breakpoint connected to the parent node is the first breakpoint of the current node, but the first breakpoint and the last breakpoint are chemically equivalent, then the identifiers can be omitted by traversing the breakpoints on the node in reverse order. We apply the above rules to Supplementary Algorithms 1 and 2. In this case, the block sequence of the example molecule Adagrasib is 7 → 3 → 2 → 1 → _ → 8 → 4 → 5 → 6, with a length of only 9. Node 5 satisfies condition 2, so the breakpoints of this node are traversed in reverse order. Supplementary Fig. 16 represents our method validation on all molecules in ChEMBL, illustrating that our method has reduced the sequence lengths by nearly half and remains effective and efficient.

The architecture of DeepBlock

In DeepBlock, the generative process for block B based on the target protein T can be represented as the conditional distribution $p(B|T) = \int p(B|z, T)p(z|T)\mathrm{d}z$, where z represents the latent vector. Our goal is to approximate $p(B|z, T)$ and $p(z|T)$ using deep neural networks parameterized by θ, denoted as $p_\theta(B|z, T)$ and $p_\theta(z|T)$. In our model, as shown in Fig. 1b, $p_\theta(z|T)$ corresponds to the prior network, and $p_\theta(B|z, T)$ corresponds to the blocks decoder. During the molecular generation stage (Fig. 1c), we first sample a latent vector z from the prior network $p_\theta(z|T)$ conditioned on T, and then use $p_\theta(B|z, T)$ to generate B. To train the prior network $p_\theta(z|T)$, we introduce an recognition network parameterized by ϕ, denoted as $q_\phi(z|B, T)$, which approximates the true posterior distribution $p(z|B, T)$. We then minimize the KL divergence[32,33] between $q_\phi(z|B, T) \approx \mathcal{N}(\mu, \sigma^2\mathbf{I})$ and $p_\theta(z|T) \approx \mathcal{N}(\mu', \sigma'^2\mathbf{I})$, where μ and μ' represent the means, σ^2 and σ'^2 represent the variances, and $\mathbf{I}$ is the identity matrix. The reparameterization trick[33] is used to stochastically sample a latent vector z from the latent space $\mathcal{N}(\mu, \sigma^2\mathbf{I})$ or $\mathcal{N}(\mu', \sigma'^2\mathbf{I})$, while preserving the differentiability of the model.

The blocks encoder and target encoder of BGNet are used to encode the ligand block sequence and protein sequence into the vectors. Furthermore, we designed a contribution perception network to predict the contribution score for each residue. The contribution score is inversely related to the distance between the residue and the center of the target binding pocket. In other words, residues closer to the target pocket center receive higher target contribution scores, indicating their greater relevance in ligand binding. These target contribution scores are utilized to weight the summation of protein residue features learned by the target encoder, ultimately yielding the target representation.

Blocks encoder and decoder. The blocks encoder $\mathrm{Encoder}_B$, based on gate recurrent unit (GRU)[34], takes the ligand molecular blocks sequence $\mathbf{B}$ and obtains the hidden states from each layer of the GRU.

These hidden states are concatenated to form the ligand representation $x = \mathrm{GRU}(\mathbf{B})$. The ligand decoder $\mathrm{Decoder_B}$, is also based on GRU. Given the latent vectors z or z' and the target representation c, it reconstructs the sequence $\hat{\mathbf{B}} = \mathrm{GRU}(W^{(c)}[z\,c])$. Here $[z\,c]$ represents the concatenation of z and c. $W^{(c)}$ denotes the weight matrix used for adjusting dimensions before passing $[z\,c]$ through the ligand GRU decoder.

In addition, we incorporate a BOW loss[32] into the ligand decoder to address the issue of vanishing latent variable in conditional variational autoencoder (CVAE). The BOW loss decomposes the target sequence representation x into two variables: one that preserves the word order, denoted as x_{ORD}, and another that disregards the word order, denoted as x_{BOW}.

Assuming that x_{ORD} and x_{BOW} are conditionally independent given z and c, we have $p(x, z|c) = p(x_{\mathrm{ORD}}|z, c)p(x_{\mathrm{BOW}}|z, c)p(z|c)$. To approximate $p(x_{\mathrm{BOW}}|z, c)$, we utilize a neural network and minimize the negative log-likelihood $\log p(x_{\mathrm{BOW}}|z, c)$. By doing so, the latent variable z captures the global information of the target sequence. The approximation is given by the following equation, where MLP represents multilayer perceptron:

$$\hat{\mathbf{P}} = p(x_{\mathrm{BOW}}|z, c) = \mathrm{Softmax}(\mathrm{MLP_{BOW}}([z\,c])) \tag{1}$$

Binding contribution of the residue. In protein–ligand interaction studies, the distance between ligands and residues in the protein often affects their interactions and binding affinity. In general, when the distance between a residue and a ligand molecule is large, their interaction is weakened and the binding affinity is reduced. Therefore, we designed a coefficient α_i called the residue's binding contribution, which is negatively correlated with the distance from the target pocket center to each residue.

$$\alpha_i = \mathrm{Softmax}\left(-\left(\frac{d_i}{k}\right)^{\tau}\right) \tag{2}$$

Here $k = 20$, $\tau = 2.5$ and d_i represents the distance from the centroid of residue i to the center of the binding site. The centroid of the reference ligand molecule in the dataset is used as the pocket center, and the distances from all residue centroids to the pocket center are calculated (in ångströms; refer to Supplementary Fig. 17).

We use a multilayer perceptron network to predict the residue's binding contribution for each residue. As ligand molecules bind only to favorable binding sites in protein structures, we can rely more on the residues near the binding site to generate ligand molecules. So, in the target encoder, the residue features are then weighted and averaged using equation (3) to obtain the target feature c. The model minimizes the error between the predicted values and the actual values using the loss function in equation (7).

The residue's binding contribution serves as a auxiliary training scheme in this study and has several advantages. The independent design of predicted and actual values retains the model's ability to autonomously perceive interactions between ligand residues. Even without actual values, the model theoretically discovers the pattern of contribution even in the absence of 3D structures in the training data. During the sampling phase, the model does not require additional structural information and only needs the protein sequence as input.

Target encoder. Let the amino acid sequence be $\mathbf{T} = (t_1, t_2, \ldots, t_m)$, where t_i denotes the type of the ith amino acid in the sequence from the 20 standard amino acids, and m is the total number of amino acids. The target encoder $\mathrm{Encoder_T}$ will compute the target representation c through the process illustrated in Supplementary Fig. 18.

$$\begin{aligned} \mathbf{T}' &= \mathrm{Transformer_T}(\mathbf{T}) \\ \hat{\boldsymbol{\alpha}} &= \mathrm{Softmax}\left(\mathrm{MLP}_{\alpha}\left(\mathbf{T}'\right)\right) \\ c &= \sum_{i=1}^{m} \hat{\alpha}_i \mathbf{t}'_i \end{aligned} \tag{3}$$

First, the amino acid feature sequence $\mathbf{T}' = (\mathbf{t}'_1, \mathbf{t}'_2, \ldots, \mathbf{t}'_m)$ is extracted using the transformer neural network $\mathrm{Transformer_T}$. Then, the contribution perceptron MLP_{α}, based on MLP, evaluates the contribution of each residue individually. The contributions are normalized using the Softmax function, resulting in a contribution vector $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_m)$. Finally, the feature sequence is average-pooled using the contribution weights to obtain the target representation c.

For $\mathrm{Transformer_T}$, we use the protein pretrained language model ESM-2[21], which uses a multilayer transformer neural network and relative position encoding technique. It is pretrained using the masked language model strategy[35] and outperforms other tested single-sequence protein language models in various protein structure prediction tasks.

BGNet training

Loss function. We designed the following four loss functions for the entire DeepBlock framework. During the pretraining and training phases, different types of loss function are combined to optimize the model.

Reconstruction loss. We use cross-entropy to measure the reconstruction loss between the input ligand block sequence $\mathbf{B}$ and the model's reconstructed sequence $\hat{\mathbf{B}}$. Assuming the sequence length is K, y_k represents the one-hot encoded vector of the kth block in the sequence $\mathbf{B}$, $\hat{y}_k$ represents the predicted probability distribution vector of the kth block in the sequence $\hat{\mathbf{B}}$, and C represents the number of categories:

$$\mathcal{L}_{\mathrm{Recon}}(y, \hat{y}) = -\sum_{t=1}^{K}\sum_{i=1}^{C} y_{k,i} \log(\hat{y}_{k,i}) \tag{4}$$

BOW loss. BOW is essentially a multi-classifier, and we use cross-entropy to calculate its loss. In this case, p represents the multi-hot encoded vector of the block categories present in the actual sequence $\mathbf{B}$, and $\hat{p}$ represents the predicted probability distribution vector of $\hat{\mathbf{P}}$. C represents the number of categories:

$$\mathcal{L}_{\mathrm{BOW}}(p, \hat{p}) = -\sum_{i=1}^{C} p_i \log(\hat{p}_i) \tag{5}$$

Latent space distribution loss. To enable the prior network to learn ligand molecular information, we need to minimize the difference between the predicted probability distribution $q_{\phi}(z|x, c) \approx \mathcal{N}(\mu, \sigma^2\mathbf{I})$ by the recognition network and the predicted probability distribution $p_{\theta}(z|c) \approx \mathcal{N}(\mu', \sigma'^2\mathbf{I})$ by the prior network:

$$\mathcal{L}_{\mathrm{Latent}}(q_{\phi}, p_{\theta}) = D_{\mathrm{KL}}(q_{\phi}(z|x, c) \| p_{\theta}(z|c)) \tag{6}$$

Contribution distribution loss. We utilize discrete KL divergence to minimize the difference between the predicted target contribution values and the actual values, guiding the model to learn the integration of binding site information:

$$\mathcal{L}_{\mathrm{Contrib}}(\alpha, \hat{\alpha}) = \sum_{i=1}^{C} \alpha_i \log\left(\frac{\alpha_i}{\hat{\alpha}_i}\right) \tag{7}$$

Pretraining. To boost the performance, we incorporate the self-supervised pretraining strategies into DeepBlock. First, we pretrained the blocks autoencoder, which consists of blocks encoder, recognition network and blocks decoder in BGNet using the ChEMBL dataset[23], as shown in Fig. 1a. By exposing the model to a diverse range of chemical structures and information from the large dataset, we equip it with a broader understanding of molecular interactions and structures.

Second, we harnessed a pretrained protein language model known as ESM-2[21]. ESM-2 is a transformer neural network that has been pretrained with the mask strategy[35] on large-scale protein sequence dataset. It has shown superior performance compared with other protein

language models in various protein structure prediction tasks. By utilizing ESM-2, we extract rich and informative features from protein sequences. These features offer valuable insights into the characteristics and attributes of protein, which are essential for ligand design.

During this stage, we set $c = \mathbf{0}$ and optimize only the ligand encoder, recognition network, and the ligand decoder (including the BOW multi-classifier). The loss function is represented as follows:

$$\mathcal{L}_{\text{Pretrain}} = \mathcal{L}_{\text{Recon}} + w_1\mathcal{L}_{\text{BOW}} + w_2\mathcal{L}_{\text{Latent}} \tag{8}$$

Here the latent space distribution loss $\mathcal{L}_{\text{Latent}}$ is modified to minimize the difference between the predicted probability distribution $q_\phi(z|x,c) \approx \mathcal{N}(\mu, \sigma^2\mathbf{I})$ by the inference network and the standard normal distribution $\mathcal{N}(0,\mathbf{I})$:

$$\mathcal{L}_{\text{Latent}}(q_\phi, p_\theta) = D_{\text{KL}}(q_\phi(z|x,c)\|\mathcal{N}(0,\mathbf{I})) \tag{9}$$

Here $D_{\text{KL}}(P||Q)$ denotes the KL divergence between the distributions P and Q. The generative model at this stage is essentially a VAE structure. We use the Adam optimizer[36] to optimize $\mathcal{L}_{\text{Pretrain}}$. The learning rate decays with the training rounds. We use KL annealing[37] to adjust the value of w_2, starting from 0 in the first round and gradually increasing it to 1 in the later stages. w_1 is set to 1. After training for 200 rounds, we select the model parameter with the smallest overall loss $\mathcal{L}_{\text{Pretrain}}$ on the validation set as the starting parameters for the subsequent training phase.

Training. The pipeline for formal training is shown in Fig. 1b, and it utilizes the protein–ligand pairs dataset CrossDocked 2020. We use the Adam optimizer[36] to minimize the following loss function:

$$\mathcal{L} = \mathcal{L}_{\text{Recon}} + w_1\mathcal{L}_{\text{BOW}} + w_2\mathcal{L}_{\text{Latent}} + w_3\mathcal{L}_{\text{Contrib}} \tag{10}$$

The annealing method for w_2 is the same as in the pretraining phase, where $w_1 = w_3 = 1$. We select the model parameter after training for 200 rounds as the final test benchmark. At this point, the target contribution encoder is able to capture binding site information and encode target representations. The prior network is able to predict the ligand molecular latent space from target information for subsequent sampling and reconstruction operations.

The parameter size of the ESM-2 ranges from 8 million to 15 billion. To improve the training efficiency, we downloaded the pretrained model parameters and used them solely for inference. In addition, we designed a caching strategy to accelerate the multiple rounds training.

Molecular generation

The pipeline for molecular generation is shown in Fig. 1c. It requires only the input of the target protein amino acid sequence. After passing through the target encoder (including the target contribution perception) and the prior network, the predicted ligand molecular latent space $\mathcal{N}(\mu', \sigma'^2\mathbf{I})$ is obtained. Multiple random samples are taken from the latent space, and diverse targeted ligand molecular sequences can be obtained by decoding them using the ligand decoder. Then, following the method described in 'From fragment sequence to molecule', the sequences are reconstructed into molecules. The reconstructed molecules are checked for legality, uniqueness and the ability to generate the correct initial 3D conformation[38]. This process is repeated until the required number of molecules is collected.

Here we evaluate the performance of DeepBlock in targeted molecule generation and compare it with the baseline in multiple dimensions. We use the CrossDocked dataset[28] and follow the same filtering and splitting strategies as in previous work[16], which results in 100,000 protein–ligand pairs for the training set and 100 proteins for the test set. Using the trained model, we generate 100 different molecules for each protein in the test set for testing. The known ligands for proteins in the test set are regarded as reference ligands. Here we compared the quality of the generated molecules with following baseline models: (1) LiGAN[15] based on 3D convolutional neural networks, (2) 3D-SBDD[16] based on graph neural networks and autoregressive generation, (3) Pocket2Mol based on E(3)-equivariant generative network in an autoregressive way, and (4) TargetDiff based on 3D equivariant diffusion model. Details of metrics can be found in 'Metrics'.

Molecular property control

Here we use DeepBlock for binding affinity optimization ('Binding affinity optimization') and target-aware molecular optimization ('Target-aware molecular optimization'). Principally, when a molecule and a protein target are fed into DeepBlock, and a latent vector is resampled, the model effectively generates a novel molecule. This molecule, noteworthy for its similarity to the original compound and its capability to bind with the protein target, is appropriately deemed a neighbor of the original.

Thereby, we used BGNet as the candidate molecule generator in the SA algorithm. Then we defined an objective function that encapsulates the specific properties for optimization. In each iteration, we input the existing molecule and the relevant target into BGNet to produce a new molecule. The decision of whether to adopt this novel molecule is guided by the Metropolis rule[39,40], which considers the energy change associated with this transition. In this way, we can control the model to generate molecules with desired properties for a given protein target.

Binding affinity optimization. By inputting a protein target and a molecule with poor affinity into the trained DeepBlock, the model can generate new molecules that show similarities to the original compound while manifesting an improved affinity for the protein target. We evaluated the binding affinity optimization capabilities of DeepBlock by selecting 'F16P1 (3kc1)' from the CrossDocked 2020 test set as the target receptor and randomly choosing 5,000 small molecules from the ChEMBL dataset as the initial molecules for optimization. The initial molecules were input into BGNet as shown in Supplementary Fig. 19, where their latent vectors were resampled and decoded into new molecular building blocks, which were then reconstructed into optimized molecules. We evaluated the similarity of these pre- and post-optimized molecules using the Tanimoto fingerprint[41], and compared the changes in docking affinity.

Target-aware molecular optimization. Supplementary Algorithm 3 implements SA-based target-aware molecular optimization. The initial molecule x_0 can either be generated by the model through direct sampling based on the target or can be arbitrarily specified as an existing molecule. Next, OPTIMIZETARGET(x, c) represents the process described in 'Binding affinity optimization', where the current molecule x and target c are used to optimize or expand the molecular latent space. This process involves randomly sampling new molecules as perturbed neighboring molecules x'. If the energy of the new molecule x' is lower than that of the current molecule x, the perturbation is accepted. Otherwise, the perturbation is accepted or rejected with a variable probability P. The molecule with the lowest energy throughout this process is selected as the optimal molecule, denoted as x^*.

For the energy function $E(\cdot)$, we calculate it by weighting the quantitative estimate of drug likeness QED(x) and drug toxicity TOX(x). This weighting is used to avoid generating molecules that have low toxicity but overly simplistic structures, which may fall outside the chemical space recognized by the model. In addition, linear transformations are applied to align the directions of energy minimization for these two parameters.

$$E(x) = w_{\text{TOX}}\,(0.5\text{TOX}(x) - 0.5) + w_{\text{QED}}\,(-\text{QED}(x) + 1) \tag{11}$$

In the implementation of BO, we similarly used the energy function as the objective function. We encoded the initial molecules through the recognition network, taking μ as the latent vector z, and used the Gaussian process to fit the latent vectors with the energy function, as shown in Supplementary Fig. 20. We performed the BO loop with expected improvement, generating new candidates in each iteration to be added to the Gaussian process training samples.

We used the small-mouse intraperitoneal LD_{50} (median lethal dose) sub-dataset from TOXRIC (acute toxicity)[42–44] in a dimensionless version to train the toxicity predictor TOX(x). The MACCS (Molecular Access System) molecular fingerprint[45] was used as the feature representation. We utilized the automated machine learning tool TPOT to train and select the optimal regression model and hyperparameters[46] for quantitatively estimating the drug toxicity of molecules. The regression performance on the test set is as follows: RMSE (root mean squared error) = 0.4326 and R^2 = 0.6029, which outperforms the benchmark provided by TOXRIC project[44].

To assess the molecular optimization capability of DeepBlock, we utilized the model based on the target receptor 'CCPR (1a2g)'[47] to generate 100 different initial molecules x_0. The energy function weights in equation (11) were set as w_{TOX} = 0.6 and w_{QED} = 0.4. Each molecule underwent N = 50 optimization iterations using Supplementary Algorithm 3. We also tested molecular optimization using BO with different numbers of candidates (50, 100 and 200) during expected improvement. As BO cannot perform independent optimization for each molecule, we instead selected the top-100 molecules from each iteration for analysis. We analyzed the changes in various molecular properties during the optimization process and examined the optimization results.

Metrics

(1) We use QuickVina2 (ref. 48) to compute the Vina score[49,50], which is a physics-based estimation of binding affinity between the molecules and their target. For the SMILES generated by DeepBlock, we use the ETKDG algorithm[38] to generate the initial 3D conformation of the molecules. (2) QED is used for quantitative estimation of drug likeness, which measures the likelihood of a molecule becoming a drug candidate[51]. Compounds with scores greater than 0.5 are generally considered to have potential as drug molecules. (3) Synthetic accessibility (SA) is used to evaluate the difficulty of synthesizing compounds. In this paper, we use the SAscore[52], where a lower score indicates that the compound is easier to synthesize. (4) log$P$ reflects the distribution of a substance in the oil–water phases[53]. (5) We use the retro-synthesis tool Retro*[54] to predict the synthesis route of the generated molecules and calculate the proportion of successful predictions. (6) The average similarity of Tanimoto fingerprints calculated from pairwise combinations of molecules generated from the same protein[41] is computed to measure the diversity of the model-generated molecules, which reflects the chemical space of the model. (7) Fragment similarity compares distributions of BRICS fragments in generated and reference sets. (8) Scaffold similarity compares the frequencies of Bemis–Murcko scaffolds[55] in generated and reference sets. (9) We computed the average time required for each strategy to generate 100 molecules.

Molecular dynamics simulation

After obtaining the structure of the protein–ligand complex, we first used the Protein Preparation Workflow module in Maestro to preprocess the protein, preventing incorrect structures from causing molecular simulation crashes. All molecular dynamics simulations were performed using GROMACS 2023.2. The antechamber program generated each ligand's force field parameters. The GAFF2 and AMBER14SB force fields were used for the ligands and protein, respectively. TIP3P was adopted to parametrize water. Following a 500 ps energy minimization and 1 ns restrained dynamics simulation, a 100 ns conventional molecular dynamics simulation was conducted. The stability of the ligand binding to the protein was determined by calculating the ligand's RMSD from the conventional molecular dynamics simulation trajectory. Finally, interaction analysis of the simulation trajectories was completed using the Python packages ProLIF and MDAnalysis.

Statistics and reproducibility

The data analyses were conducted with Python 3.10, pytorch 1.12.1, rdkit 2020.09.5 and QuickVina2. No data were excluded from the analysis. $P < 0.05$ was considered statistically significant. No statistical methods were used to pre-determine sample sizes. The experiments and data analyses were done randomized and the investigators were blinded to allocation during experiments and outcome assessment. To ensure the reproducibility of the findings, we report the statistics in the legend and text of each experiment.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

Source data for Figs. 2–6 are provided with this paper. All of the datasets used in this study are publicly available. The raw data of the CrossDocked 2020 dataset were obtained from https://github.com/gnina/models/tree/master/data/CrossDocked2020. The dataset for pretraining BGNet were obtained from ChEMBL dataset (https://ftp.ebi.ac.uk/pub/databases/chembl/ChEMBLdb/releases/chembl_31/). The processed datasets used to train the model are available via figshare[56] at https://figshare.com/articles/dataset/crossdocked_pocket10_with_protein_tar_gz/25878871. The small-mouse intraperitoneal LD_{50} sub-dataset was obtained from TOXRIC (https://toxric.bioinforai.tech/home).

Code availability

The source code and weights of trained models are available on GitHub at https://github.com/BioChemAI/DeepBlock and deposited on Zenodo at https://doi.org/10.5281/zenodo.13852436 (ref. 57).

References

1. Shoichet, B. K. Virtual screening of chemical libraries. *Nature* **432**, 862–865 (2004).
2. Meyers, J., Fabian, B. & Brown, N. De novo molecular design and generative models. *Drug Discov. Today* **26**, 2707–2715 (2021).
3. Wang, M. et al. Deep learning approaches for de novo drug design: an overview. *Curr. Opin. Struc. Biol.* **72**, 135–144 (2022).
4. Segler, M. H., Kogej, T., Tyrchan, C. & Waller, M. P. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Cent. Sci.* **4**, 120–131 (2018).
5. Moret, M., Friedrich, L., Grisoni, F., Merk, D. & Schneider, G. Generative molecular design in low data regimes. *Nat. Mach. Intell.* **2**, 171–180 (2020).
6. Gómez-Bombarelli, R. et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* **4**, 268–276 (2018).
7. Jin, W., Barzilay, R. & Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *Proc. 35th International Conference on Machine Learning*, *Proc. Machine Learning Research* Vol. 80 (eds Dy, J. & Krause, A.) 2323–2332 (PMLR, 2018).
8. Li, Y., Zhang, L. & Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *J. Cheminform.* **10**, 33 (2018).
9. Putin, E. et al. Reinforced adversarial neural computer for de novo molecular design. *J. Chem. Inf. Model.* **58**, 1194–1204 (2018).
10. Zang, C. & Wang, F. Moflow: an invertible flow model for generating molecular graphs. In *Proc. 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* 617–626 (ACM, 2020).

11. Kuznetsov, M. & Polykovskiy, D. MolGrow: a graph normalizing flow for hierarchical molecular generation. *Proc. AAAI Conf. Artif. Intell.* **35**, 8226–8234 (2021).
12. Hoogeboom, E., Satorras, V. G., Vignac, C. & Welling, M. Equivariant diffusion for molecule generation in 3D. In *Proc. 39th International Conference on Machine Learning*, *Proc. Machine Learning Research* Vol. 162 (eds Chaudhuri, K. et al.) 8867–8887 (PMLR, 2022).
13. Schneuing, A. et al. Structure-based drug design with equivariant diffusion models. Preprint at https://arxiv.org/abs/2210.13695 (2022).
14. Li, J. et al. Mining for potent inhibitors through artificial intelligence and physics: a unified methodology for ligand based and structure based drug design. *J. Chem. Inf. Model.* https://doi.org/10.1021/acs.jcim.4c00634 (2024).
15. Ragoza, M., Masuda, T. & Koes, D. R. Generating 3D molecules conditional on receptor binding sites with deep generative models. *Chem. Sci.* **13**, 2701–2713 (2022).
16. Luo, S., Guan, J., Ma, J. & Peng, J. A 3D generative model for structure-based drug design. *Adv. Neural Inf. Process. Syst.* **34**, 6229–6239 (2021).
17. Gao, W. & Coley, C. W. The synthesizability of molecules proposed by generative models. *J. Chem. Inf. Model.* **60**, 5714–5723 (2020).
18. Brenner, S. & Lerner, R. A. Encoded combinatorial chemistry. *Proc. Natl Acad. Sci. USA* **89**, 5381–5383 (1992).
19. Liu, R., Li, X. & Lam, K. S. Combinatorial chemistry in drug discovery. *Curr. Opin. Chem. Biol.* **38**, 117–126 (2017).
20. Bertsimas, D. & Tsitsiklis, J. Simulated annealing. *Stat. Sci.* **8**, 10–15 (1993).
21. Lin, Z. et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379**, 1123–1130 (2023).
22. Degen, J., Wegscheid-Gerlach, C., Zaliani, A. & Rarey, M. On the art of compiling and using 'drug-like' chemical fragment spaces. *ChemMedChem* **3**, 1503–1507 (2008).
23. Mendez, D. et al. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Res.* **47**, D930–D940 (2019).
24. Wang, R., Fang, X., Lu, Y. & Wang, S. The PDBbind database: collection of binding affinities for protein–ligand complexes with known three-dimensional structures. *J. Med. Chem.* **47**, 2977–2980 (2004).
25. Jessani, N., Liu, Y., Humphrey, M. & Cravatt, B. F. Enzyme activity profiles of the secreted and membrane proteome that depict cancer cell invasiveness. *Proc. Natl Acad. Sci. USA* **99**, 10335–10340 (2002).
26. Chiang, K. P., Niessen, S., Saghatelian, A. & Cravatt, B. F. An enzyme that regulates ether lipid signaling pathways in cancer annotated by multidimensional profiling. *Chem. Biol.* **13**, 1041–1050 (2006).
27. Chang, J. W., Nomura, D. K. & Cravatt, B. F. A potent and selective inhibitor of KIAA1363/AADACL1 that impairs prostate cancer pathogenesis. *Chem. Biol.* **18**, 476–484 (2011).
28. Francoeur, P. G. et al. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *J. Chem. Inf. Model.* **60**, 4200–4215 (2020).
29. Steinegger, M. & Söding, J. mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028 (2017).
30. Jänne, P. et al. KRYSTAL-1: activity and safety of adagrasib (MRTX849) in advanced/metastatic non-small cell lung cancer (NSCLC) harboring $KRAS^{G12C}$ mutation. *Eur. J. Cancer* **138**, S1–S2 (2020).
31. Landrum, G. RDKit: open-source cheminformatics. *RDKit* http://www.rdkit.org (2006).
32. Zhao, T., Zhao, R. & Eskenazi, M. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proc. 55th Annual Meeting of the Association for Computational Linguistics* Vol. 1 (eds Barzilay, R. & Kan, M.) 654–664 (ACL, 2017).
33. Kingma, D. P. & Welling, M. Auto-encoding variational Bayes. Preprint at https://arxiv.org/abs/1312.6114 (2014).
34. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. Preprint at https://arxiv.org/abs/1412.3555 (2014).
35. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. North American Chapter of the Association for Computational Linguistics* Vol. 1 (eds Burstein, J. et al.) 4171–4186 (ACL, 2019).
36. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. Preprint at https://arxiv.org/abs/1412.6980 (2015).
37. Bowman, S. R. et al. Generating sentences from a continuous space. In *Proc. 20th SIGNLL Conference on Computational Natural Language Learning* (eds Riezler, S. & Goldberg, Y.) 10–21 (ACL, 2016).
38. Riniker, S. & Landrum, G. A. Better informed distance geometry: using what we know to improve conformation generation. *J. Chem. Inf. Model.* **55**, 2562–2574 (2015).
39. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087–1092 (1953).
40. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
41. Bajusz, D., Rácz, A. & Héberger, K. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *J. Cheminform.* **7**, 20 (2015).
42. Jain, S. et al. Large-scale modeling of multispecies acute toxicity end points using consensus of multitask deep learning methods. *J. Chem. Inf. Model.* **61**, 653–663 (2021).
43. Liwanag, P. M., Hudson, V. W. & Hazard, G. F. Jr. ChemIDplus: a web-based chemical search system. *NLM* https://www.nlm.nih.gov/pubs/techbull/ma00/ma00_chemid.html (2000).
44. Wu, L. et al. TOXRIC: a comprehensive database of toxicological data and benchmarks. *Nucleic Acids Res.* **51**, D1432–D1445 (2023).
45. Durant, J. L., Leland, B. A., Henry, D. R. & Nourse, J. G. Reoptimization of mdl keys for use in drug discovery. *J. Chem. Inf. Comput. Sci.* **42**, 1273–1280 (2002).
46. Le, T. T., Fu, W. & Moore, J. H. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics* **36**, 250–256 (2020).
47. Cao, Y., Goodin, D. & Mcree, D. Probing the strength and character of an Asp-His-x hydrogen bond by introducing buried charges. *PDB* https://doi.org/10.2210/pdb1a2g/pdb (1998).
48. Alhossary, A., Handoko, S. D., Mu, Y. & Kwoh, C.-K. Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics* **31**, 2214–2216 (2015).
49. Trott, O. & Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J. Comput. Chem.* **31**, 455–461 (2010).
50. Eberhardt, J., Santos-Martins, D., Tillack, A. F. & Forli, S. AutoDock Vina 1.2. 0: new docking methods, expanded force field, and Python bindings. *J. Chem. Inf. Model.* **61**, 3891–3898 (2021).
51. 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).
52. Ertl, P. & Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J. Cheminform.* **1**, 8 (2009).

53. Wildman, S. A. & Crippen, G. M. Prediction of physicochemical parameters by atomic contributions. *J. Chem. Inf. Comput. Sci.* **39**, 868–873 (1999).
54. Chen, B., Li, C., Dai, H. & Song, L. Retro*: learning retrosynthetic planning with neural guided A* search. In *Proc. 37th International Conference on Machine Learning* Vol. 119 (eds Daumé, H. III & Singh, A.) 1608–1616 (PMLR, 2020).
55. Bemis, G. W. & Murcko, M. A. The properties of known drugs. 1. Molecular frameworks. *J. Med. Chem.* **39**, 2887–2893 (1996).
56. Zhang, K. & Li, P. crossdocked_pocket10_with_protein.tar.gz. *figshare* https://figshare.com/articles/dataset/crossdocked_pocket10_with_protein_tar_gz/25878871 (2024).
57. Li, P. & Zhang, K. Biochemai/deepblock. *Zenodo* https://doi.org/10.5281/zenodo.13852436 (2024).

Acknowledgements

This work was supported by the National Science and Technology Major Project (2023ZD0120902 to X.Z.), the National Natural Science Foundation of China (62202353 to P.L.; U22A2037 to X.Z.; 62425204 to X.Z.; 62122025 to X.Z.; 62450002 to X.Z.; and 62432011 to X.Z.).

Author contributions

P.L. and X.Z. conceived the research project. P.L. and K.Z. designed and implemented the framework. P.L., X.Y., L.G. and X.Z. designed the experiments. P.L., K.Z., T.L., Y.C., X.Y., L.G. and X.Z. conducted the experiments and results analyses. R.L. conducted the molecular dynamics simulation. All the authors discussed the experimental results and commented on the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s43588-024-00718-0.

Correspondence and requests for materials should be addressed to Xiangxiang Zeng.

Peer review information *Nature Computational Science* thanks Kil To Chong and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editor: Kaitlin McCardle, in collaboration with the *Nature Computational Science* team.

nature portfolio

Corresponding author(s): Xiangxiang Zeng

Last updated by author(s): Sep 29, 2024

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☐	☒ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☒	☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☐	☒ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	No software was used.
Data analysis	We used open-source software, including python 3.10, pytorch 1.12.1, rdkit 2020.09.5, and QuickVina2, for data analysis.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

All of the datasets used in this study are publicly available. Raw and processed data of CrossDocked 2020 dataset were obtained from https://github.com/luost26/3D-Generative-SBDD/tree/main. The dataset for pre-training BGNet were obtained from ChEMBL dataset (https://www.ebi.ac.uk/chembl/). The processed datasets used to train the model are available via Figshare (https://figshare.com/articles/dataset/crossdocked_pocket10_with_protein_tar_gz/25878871). The small-mouse intraperitoneal LD50 sub-dataset were obtained from TOXRIC (https://toxric.bioinforai.tech/home).

Research involving human participants, their data, or biological material

Policy information about studies with human participants or human data. See also policy information about sex, gender (identity/presentation), and sexual orientation and race, ethnicity and racism.

Reporting on sex and gender	N/A
Reporting on race, ethnicity, or other socially relevant groupings	N/A
Population characteristics	N/A
Recruitment	N/A
Ethics oversight	N/A

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	We did not perform a sample size calculation, as the sample sizes were determined based on the availability of datasets and aligned with conventions within the field of computational studies utilizing these datasets. This approach ensures that our findings are reliable and comparable to existing research.
Data exclusions	No data were excluded.
Replication	In our study, we conducted 100 independent generation experiments for each test target using the model to evaluate molecular generation. Additionally, we performed 100 independent optimization experiments for molecular optimization. All attempts at replication were successful.
Randomization	In this study, samples were allocated into experimental groups through random selection, Allocation was entirely random, and covariate control was not relevant to this computational study focusing on public datasets.
Blinding	Blinding was not relevant to this study. As it is a computational analysis using publicly available datasets, there was no group allocation or subjective assessment that required blinding during data collection or analysis, The datasets were used as provided, with predefined structures and content, ensuring objective and unbiased data processing.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern
☒	☐ Plants

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Plants

Seed stocks	N/A
Novel plant genotypes	N/A
Authentication	N/A