Fine-tuning Pocket-conditioned 3D Molecule Generation via Reinforcement Learning

Daeseok Lee[*], **Yongjun Cho**[*]
Deargen Inc.
Seoul, South Korea
{daelee, cyjun0304}@deargen.me

Abstract

In drug discovery, generating molecules that bind to target proteins while possessing desired chemical properties is a fundamental challenge. In the early stages of deep learning for molecule generation, most models did not explicitly model the interactions between the generated molecules and the target protein. Instead, they generated molecules as SMILES strings or graphs and considered target proteins only through docking scores. This approach faced limitations in generalization and required retraining of the model for different targets. In contrast, the novel pocket-conditioned 3D molecule generation approach demonstrates significant improvements by generating 3D molecular structures from protein binding pocket inputs. For example, Pocket2Mol, an atom-autoregressive model, shows superior performance in creating molecules with high binding affinity and desirable chemical properties. However, it has limitations, including the production of unrealistic stereochemical structures and constraints on the properties of generated molecules, the latter stemming from its training to replicate the molecules in the training set. To overcome these limitations, we propose a reinforcement learning method to fine-tune the model to generate molecules with enhanced properties. To demonstrate its effectiveness, we conducted an experiment where we used our method to minimize the model's stereochemical issues and enhance the drug-likeness and binding affinity of the generated molecules. Our results show that this approach not only resolves Pocket2Mol's existing problems but also establishes new benchmarks for molecule generation metrics, highlighting our method's potential to advance molecule generation. The source code for inference and instructions on reproduction can be found at https://github.com/deargen/Pocket2Mol_RL_public.

1 Introduction

In drug discovery, identifying molecules that bind to target proteins while possessing desired chemical properties represents a fundamental challenge. Methods of molecule generation tackle this problem through the utilization of generative models. Recent advancements in deep learning have substantially enhanced the capabilities of these molecule generation models.

In the early stages of deep learning for molecule generation, most models did not explicitly model the interactions between the generated molecules and the target protein (Jin et al. (2019), Olivecrona et al. (2017), Zhou et al. (2019)). Instead, they generated molecular topology, represented either by SMILES strings or by graphs, and relied on metrics such as docking scores (Trott & Olson (2010)) to evaluate the interactions with the protein. This approach necessitated retraining the model each time the target protein changed, leading to challenges in generalizing the model's performance across various targets.

Pocket-conditioned 3D molecule generation, a new approach differing from traditional methods, overcame this challenge. This class of methods inputs the structure of binding pockets and outputs 3D molecule conformers, providing several advantages (Xie et al. (2022), Zhang et al. (2023)).

[*]These authors contributed equally to this work

Firstly, it incorporates all local interaction constraints imposed by the protein pocket during the iterative generation process, thereby mimicking strategies employed by human experts. Secondly, it predicts the molecule's binding pose within the protein pocket during generation, bypassing the need for subsequent, computationally intensive and error-prone docking algorithms to determine this pose. Lastly, the protein structure input allows for effective knowledge transfer from extensive databases of protein-ligand complexes, obviating the need for further target-specific training. For more detailed information on pocket-conditioned 3D molecule generation methods, refer to Appendix B.

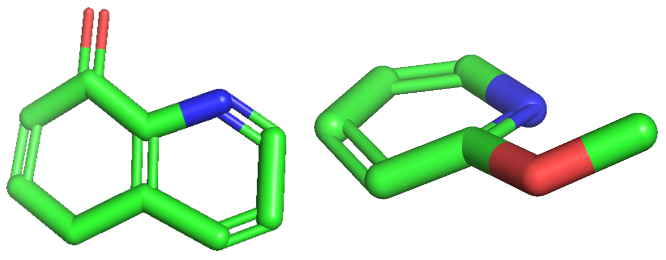

(a) A generated molecule with bond length problems (b) A generated molecule with ring flatness problems

Figure 1: Stereochemical issues in molecules generated by Pocket2Mol.

Among the existing pocket-conditioned 3D generation methods, this study extends the capabilities of an atom-autoregressive model Pocket2Mol (Peng et al. (2022)), recognized for its state-of-the-art performance. However, Pocket2Mol is not without limitations. Its atom-autoregressive approach occasionally results in unrealistic intra-molecular structures in the generated conformers. For example, Figure 1 illustrates how molecules generated by Pocket2Mol often possess non-flat ring systems, a deviation from typical molecular structures. The extent of such stereochemical issues, as identified by the criteria suggested in Buttenschoen et al. (2024), will be thoroughly investigated in Section 4. Moreover, Pocket2Mol, like other existing pocket-conditioned 3D generation methods, primarily trains to maximize the likelihood of its training data. This dependence on the training dataset suggests a potential for further model optimization, guided by computable molecular properties.

In this study, we introduce a method that utilizes reinforcement learning to overcome the limitations observed in previous models. In particular, our method focuses on fine-tuning the Pocket2Mol model to achieve two key objectives: first, to minimize stereochemical issues, and second, to enhance properties such as drug-likeness and binding affinity. Although we developed our method based on Pocket2Mol, most of its design elements and techniques might be easily adapted to enhance other pocket-conditioned 3D molecule generation models.

Our experiments demonstrate that our technique notably enhances binding affinity, drug-likeness, and synthetic accessibility, while concurrently diminishing stereochemical issues. It not only improves the Pocket2Mol model but also achieves state-of-the-art results for most of the metrics commonly used in molecule generation benchmarks. These results show that reinforcement learning is a potent tool not only for rectifying the issues in pocket-conditioned 3D molecule generation models but also for significantly enhancing them. This could play a crucial role in addressing the challenge of molecule generation.

2 POCKET2MOL

Our reinforcement learning method relies on the pre-trained Pocket2Mol (Peng et al. (2022)) model. Its neural network architecture comprises specialized sub-modules, each tasked with predicting specific aspects of generated atoms: frontier atoms, atom coordinates, atom types, and bond types. These modules undergo supervised learning using ground truth samples derived from partially masked molecules in the training dataset. During inference, Pocket2Mol employs a sampling algorithm involving a beam search. This process involves oversampling potential atom configurations—combinations of a frontier atom, atom coordinate, atom type, and bond types—which are then filtered and prioritized based on thresholds and probability criteria. A comprehensive explanation of this process is provided in Appendix C.

3 METHOD

In this section, we explain the components of our reinforcement learning algorithm—state, action, sampling, reward, and optimization algorithm.

3.1 STATE AND ACTION

Since our method is based on Pocket2Mol, which is an atom-autoregressive model, the state is defined as the combination of the partially generated molecule and the atoms that make up the binding pocket. In the same manner as Pocket2Mol, we represent the partially generated molecules and the binding pockets together as a single graph. In this graph, nodes represent atoms and edges indicate spatial proximity. The node feature includes properties such as the atom types and the edge feature includes properties such as the bond types (Peng et al. (2022)).

Actions are categorized as either an *atom action* or a *stop action*. An *atom action* involves adding a new atom to the molecule being constructed and comprises four components: (1) the index of the *focal atom*, (2) the coordinate, (3) the element type of the new atom, and (4) the types of bonds formed between the new atom and the existing atoms, including the possibility of no connection. The *focal atom* is a concept from Pocket2Mol, indicating an already existing atom that serves as the branching point for the new addition. The *stop action* signals the end of the generation process, resulting in the completion of the molecule based on the last state's graph configuration if feasible.

Formally, an atom action is represented by the expression: $(\mathbf{f}, \mathbf{p}, \mathbf{a}, \mathbf{b}) \in \{1, \cdots, n_{atoms}\} \times \mathbb{R}^3 \times \{C, N, O, F, P, S, Cl\} \times \{\cdot, -, =, \equiv\}^{n_{atoms}}$. A stop action is denoted as STOPACTION.

3.2 SAMPLING DURING TRAINING

To effectively implement reinforcement learning, it is necessary to modify the sampling procedure of Pocket2Mol (as detailed in Appendix C.2). The primary reason for this modification is to ensure an independent probability distribution for sampling each molecule. The original Pocket2Mol sampling procedure creates dependencies between the sampled molecules, which contradicts this requirement. Another critical requirement for the sampling procedure is the existence of a tractable probability mass/density function. The original procedure's steps, such as threshold-based filtering and probability-based prioritization, overly complicate the probability calculations, making them practically infeasible.

Consequently, we use a new sampling procedure that meets these essential requirements, defined by the following action distribution:

1. Given the focal logits $\mathbf{y}_{\text{focal}} \in \mathbb{R}^{n_{\text{atoms}}}$ produced by the frontier module, STOPACTION is returned with probability $1 - \text{sigmoid}(\max(\mathbf{y}_{\text{focal}}))$.
2. Otherwise, an atom action $(\mathbf{f}, \mathbf{p}, \mathbf{a}, \mathbf{b})$ is determined as follows:
   (a) A focal index $\mathbf{f} \in \{1, \cdots, n_{\text{atoms}}\}$ is sampled according to $\text{softmax}(\mathbf{y}_{\text{focal}})$.
   (b) A position $\mathbf{p}$ is sampled from the Gaussian mixture distribution $\mathcal{G}(\pi, \mu, \Sigma)$ predicted by the atom position module at the index $\mathbf{f}$.
   (c) Given the atom type logits $\mathbf{y}_{\text{atom}} \in \mathbb{R}^{\{C,N,O,F,P,S,Cl\}}$, an atom type $\mathbf{a}$ is sampled according to $\text{softmax}(\mathbf{y}_{\text{atom}})$.
   (d) Given the bond type logits $\mathbf{y}^{(i)}_{\text{bond}} \in \mathbb{R}^{\{\cdot,-,=,\equiv\}}$ $(i = 1, \cdots, n_{\text{atoms}})$, the bond types $\mathbf{b} = (b_1, \cdots, b_{n_{\text{atoms}}}) \in \{\cdot, -, =, \equiv\}^{n_{\text{atoms}}}$ are sampled with $b_i \sim \text{softmax}(\mathbf{y}^{(i)}_{\text{bond}})$ for each $i = 1, \cdots, n_{\text{atoms}}$.

The repeated sampling of actions and transitions results in one of the following outcomes:

- a successful episode: STOPACTION was returned, and a molecule was successfully recovered from the graph of the last state.

- a type I failed episode: StopAction was returned, but the graph of the last state did not correspond to a valid molecule.
- a type II failed episode: the rollout continued until the number of generated atoms reached a predefined threshold (50).
- a type III failed episode: A nonsensical configuration of the molecule graph was detected during the rollout, indicating an inevitable result of either type I or type II failure, leading to early termination.

The exact conditions for type I and type III failures are detailed in Appendix D.2.

3.3 Sampling during inference

For inference, the sampling procedure outlined in the previous section is further modified to eliminate unnecessary sources of randomness. Specifically, the modifications are twofold: (1) StopAction is returned deterministically, triggered when $\mathrm{sigmoid}(\max(\mathbf{y}_{\text{focal}}))$ falls below a set threshold (0.5), and (2) the covariance matrices Σ of the Gaussian mixture distribution are reduced to zero. This leads to selecting one of the mean positions, μ, based on π. It is noteworthy that the inference procedure of Pocket2Mol also employs these strategies.

3.4 Reward

The reward in our reinforcement learning framework is designed to achieve two primary objectives: (1) to minimize the occurrence of all types of failures in episodes and (2) to optimize the desired properties of molecules from successful episodes. Specifically, a failed episode (regardless of type) incurs a failure penalty, denoted as $R = FP$. On the other hand, a successful episode is assigned a reward calculated as $R = c_1P_1 + \cdots + c_kP_k$ where c_i represents a fixed coefficient and P_i denotes a *reward term*, which is a molecular property of interest.

The values for $c_1, \cdots, c_k$, and FP are determined based on the statistics of the reward terms of molecules generated out of proteins in the training set by the pre-trained Pocket2Mol model. This approach ensures a balanced consideration of the reward terms and the failure penalty, especially during the early stages of reinforcement learning when the actor model is yet to deviate significantly from the pre-trained model. The coefficients $c_1, \cdots, c_k$, and FP are defined as follows:

$$c_i \propto \alpha_i \cdot \mathrm{sign}(P_i) \cdot \mathrm{std}(P_i)^{-1} \tag{1}$$

$$FP = \mathrm{mean}(c_1P_1 + \cdots + c_kP_k) - 2 \cdot \mathrm{std}(c_1P_1 + \cdots + c_kP_k) \tag{2}$$

In equation 1, α_i is a multiplier that defaults to 1 and signifies the property's importance in training, and $\mathrm{sign}(P_i)$ is 1 if higher values of the property P_i are desirable, and -1 otherwise.

3.5 Optimization Algorithm

For the optimization algorithm for reinforcement learning, Proximal Policy Optimization (PPO) is employed (Schulman et al. (2017)). Detailed information can be found in Appendix D.1.

4 Experiment

4.1 Training

In our experiment, we employed the following reward terms for reinforcement learning: (1) Vina score without conformer optimization, (2) Quantitative Estimate of Drug-likeness (QED) score, and (3) the proportion of stereochemical features (such as bond lengths, bond angles, ring flatness values, and distances between non-covalently bound atom pairs) that fall within designated appropriate ranges. The method for extracting these stereochemical features and their "appropriate ranges" were based on the guidelines provided in Buttenschoen et al. (2024). Details on the specific training procedures, including model freezing techniques and the application of a curriculum-based training approach, are documented in Appendix D.3.

For training and evaluation purposes, our model was applied to the CrossDocked2020 dataset (Francoeur et al. (2020)), specifically utilizing the test subset that was employed in previous research (Luo et al. (2021), Peng et al. (2022), Guan et al. (2022)). In the development phase, model validation was conducted using a randomly selected subset of the training set, which was of equivalent size to the test set.

4.2 Results

In our analysis, we assessed several key metrics: binding affinity, drug-likeness, synthetic accessibility, and the existence of certain types of stereochemical issues. This evaluation encompassed not only our method but also Pocket2Mol and various baseline approaches. These baselines include grid-based (Ragoza et al. (2022)), atom-autoregressive (Luo et al. (2021), Liu et al. (2022), Peng et al. (2022)), fragment-autoregressive (Zhang et al. (2022)), and diffusion-based (Guan et al. (2022)) methods. For each target protein in the test set, we calculated the average values of these metrics for the molecules generated specifically for that protein. The aggregated results, including both average and median values of these averages, are presented in Table 1 and Table 2.

Model	Binding Affinity		High Affinity (%)		QED		SA		Diversity (%)	
	Avg.	Med.	Avg.	Med.	Avg.	Med.	Avg.	Med.	Avg.	Med.
LiGAN (Ragoza et al. (2022))	−6.36	−6.45	13.1 %	5.1%	0.39	0.42	0.59	0.58	**34.5%**	**33.4%**
AR (Luo et al. (2021))	−6.60	−6.62	29.6 %	15.5 %	0.51	0.50	0.64	0.64	29.9 %	29.1 %
GraphBP (Liu et al. (2022))	−5.13	−5.20	14.9 %	4.8%	0.43	0.43	0.48	0.48	21.6 %	21.7 %
FLAG (Zhang et al. (2022))	−6.02	−5.99	23.8 %	7.8%	0.61	0.61	0.65	0.65	22.6 %	20.3 %
TargetDiff (Guan et al. (2022))	−7.22	**-7.37**	46.7 %	39.9 %	0.48	0.49	0.58	0.57	28.3 %	28.6 %
Pocket2Mol (Peng et al. (2022))	−7.28	−6.94	46.2 %	44.4 %	0.57	0.59	0.74	0.76	26.4 %	23.3 %
Pocket2Mol-RL (Ours)	**-7.40**	−7.35	**50.7%**	**58.6%**	**0.64**	**0.67**	**0.84**	**0.84**	23.0 %	22.6 %
Reference	−7.38	−7.35	-	-	0.48	0.47	0.73	0.74	-	-

Table 1: Comparison of Performance Metrics of Generation Models

Model	has a distorted ring (%)		has an invalid bond (%)		has an invalid angle (%)		has a clash (%)		has a problem (%)	
	Avg.	Med.	Avg.	Med.	Avg.	Med.	Avg.	Med.	Avg.	Med.
LiGAN (Ragoza et al. (2022))	14.0 %	3.3%	96.9 %	100.0 %	93.0 %	100.0 %	56.4 %	62.4 %	98.5 %	100.0 %
AR (Luo et al. (2021))	33.0 %	22.1 %	6.3%	**4.1%**	18.5 %	11.1 %	20.1 %	14.0 %	53.1 %	54.0 %
GraphBP (Liu et al. (2022))	**0.9%**	**1.1%**	94.6 %	95.1 %	96.1 %	96.6 %	77.0 %	77.1 %	98.0 %	98.3 %
FLAG (Zhang et al. (2022))	2.2%	1.3%	5.5%	4.8%	31.3 %	30.4 %	17.8 %	18.6 %	34.4 %	34.2 %
TargetDiff (Guan et al. (2022))	6.9%	6.1%	**5.2%**	**4.1%**	38.3 %	39.8 %	18.7 %	18.7 %	51.0 %	53.4 %
Pocket2Mol (Peng et al. (2022))	21.5 %	18.3 %	20.3 %	16.1 %	4.0%	**1.9%**	4.4%	1.9%	38.0 %	35.3 %
Pocket2Mol-RL (Ours)	7.1%	5.6%	17.0 %	14.1 %	**3.3%**	2.0%	**1.7%**	**1.0%**	**23.8%**	**20.5%**
Reference	0.0%	0.0%	2.0%	0.0%	0.0%	0.0%	4.0%	0.0%	6.0%	0.0%

Table 2: Analysis of Stereochemical Issues in Generated Molecules

The results demonstrate that reinforcement learning successfully improved the binding affinity and drug-likeness of Pocket2Mol, while also achieving a marked reduction in stereochemical issues, particularly in the distorted ring problem. This advancement was accompanied by an improvement in synthetic accessibility, a notable achievement given that it was not directly targeted in the reward structure. It is important to mention, however, that there was a reduction in molecular diversity, a trade-off likely resulting from the fine-tuning process being heavily focused on specific metrics. While the binding affinity reached levels similar to those of a diffusion-based baseline (Guan et al. (2022)), and certain stereochemical problems remained more frequent than in some baselines, the improvements in all other evaluated metrics significantly surpassed those of the best-performing baseline.

5 Conclusion

In summary, our reinforcement learning approach has successfully improved a pocket-conditioned 3D molecule generation model across all evaluated metrics. This achievement underscores the method's capability to optimize a wide range of desired chemical and structural properties. The significance of this work lies in two key aspects: firstly, the central challenge in drug discovery is finding molecules with specific desired properties; secondly, the utility of 3D molecule generation models depends on the validity of the predicted molecule 3D conformers. Therefore, reinforcement learning techniques like ours hold immense potential to propel advancements in the field. This study represents one of the pioneering successes in applying such an approach.

References

Martin Buttenschoen, Garrett M Morris, and Charlotte M Deane. Posebusters: Ai-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science*, 2024.

Oh-Hyeon Choung, Riccardo Vianello, Marwin Segler, Nikolaus Stiefl, and José Jiménez-Luna. Learning chemical intuition from humans in the loop. *ChemRxiv*, 2023. doi: 10.26434/chemrxiv-2023-knwnv.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2023.

Paul G Francoeur, Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B Iovanisci, Ian Snyder, and David R Koes. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. *Journal of chemical information and modeling*, 60(9):4200–4215, 2020.

Jiaqi Guan, Wesley Wei Qian, Xingang Peng, Yufeng Su, Jian Peng, and Jianzhu Ma. 3d equivariant diffusion for target-aware molecule generation and affinity prediction. In *The Eleventh International Conference on Learning Representations*, 2022.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation, 2019.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2020.

Haitao Lin, Yufei Huang, Meng Liu, Xuanjing Li, Shuiwang Ji, and Stan Z Li. Diffbp: Generative diffusion of 3d molecules for target protein binding. *arXiv preprint arXiv:2211.11214*, 2022.

Meng Liu, Youzhi Luo, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. Generating 3d molecules for target protein binding. *arXiv preprint arXiv:2204.09410*, 2022.

Shitong Luo, Jiaqi Guan, Jianzhu Ma, and Jian Peng. A 3d generative model for structure-based drug design. *Advances in Neural Information Processing Systems*, 34:6229–6239, 2021.

Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, et al. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9:48, 2017.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. In *International Conference on Machine Learning*, pp. 17644–17655. PMLR, 2022.

Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Generating 3d molecules conditional on receptor binding sites with deep generative models. *Chemical science*, 13(9):2701–2713, 2022.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, 31(2):455–461, 2010.

Weixin Xie, Fanhao Wang, Yibo Li, Luhua Lai, and Jianfeng Pei. Advances and challenges in de novo drug design using three-dimensional deep generative models. *Journal of Chemical Information and Modeling*, 62(10):2269–2279, 2022.

Zaixi Zhang, Yaosen Min, Shuxin Zheng, and Qi Liu. Molecule generation for target protein binding with structural motifs. In *The Eleventh International Conference on Learning Representations*, 2022.

Zaixi Zhang, Jiaxian Yan, Qi Liu, Enhong Chen, and Marinka Zitnik. A systematic survey in geometric deep learning for structure-based drug design, 2023.

Zhenpeng Zhou, Steven Kearnes, Li Li, et al. Optimization of molecules via deep reinforcement learning. *Scientific Reports*, 9:10752, 2019.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020.

A Inspirations from RLHF

Our method is conceptually similar to the Reinforcement Learning from Human Feedback (RLHF) approach, where pre-trained language models are fine-tuned based on a reward model trained to predict human preferences (Christiano et al. (2023), Ouyang et al. (2022), Ziegler et al. (2020), Touvron et al. (2023)). Additionally, the token-autoregressive nature of language model sentence generation in RLHF is analogous to the atom-autoregressive approach in molecule generation. Drawing inspiration from these parallels, we employ the Proximal Policy Optimization (PPO) algorithm (Schulman et al. (2017)), widely used in RLHF, to guide our model's fine-tuning process.

B Pocket-conditioned 3D Molecule Generation

Existing pocket-conditioned 3D molecule generation models exhibit diversity in their representation and construction of molecules during the generation process. Grid-based methods (Ragoza et al. (2022)) involve sampling atomic densities within voxels in a cubical grid. Atom-autoregressive methods (Luo et al. (2021), Liu et al. (2022), Peng et al. (2022)) sequentially predict atoms, including their element types, coordinates, and bonds to previously generated atoms. Fragment-autoregressive methods (Zhang et al. (2022)) sequentially predict molecular fragments selected from a predefined collection. Diffusion-based methods (Lin et al. (2022), Guan et al. (2022)) employ a process that iteratively refines molecular structures, starting from a random distribution and gradually converging to a desired structure through a series of diffusion steps, guided by learned probability distributions.

C Pocket2Mol

C.1 Model Architecture

The Pocket2Mol model is designed to predict additional atoms from given protein pockets and molecular fragments. The architecture of the Pocket2Mol model comprises four main modules: an encoder, a frontier predictor, a position predictor, and an element-and-bond predictor.

- **Encoder**: The encoder inputs protein pockets and molecular fragments as a k-nearest neighbor (KNN) graph and extracts hidden representations consisting of scalar and vector features of the vertices and edges. It captures the chemical and geometric attributes of protein pockets and molecule fragments. All predictors share common embedding and encoding layers.

- **Frontier Predictor**: The frontier predictor identifies which atoms can attach new atoms, called frontier atoms. Given multiple frontier atoms, a focal atom is sampled based on model-predicted probabilities. Utilizing additional geometric vector perceptron (GVP, Jing et al. (2020)) and MLP layers, this module generates the probability of each atom being a frontier atom.
- **Position Predictor**: For a chosen focal atom (sampled from the frontier atoms during inference), the position predictor predicts parameters for a Gaussian mixture distribution. It predicts the mean, covariance, and prior probability of the Gaussian components through an additional GV-MLP block and a geometric vector linear (GVL) block.
- **Element-and-Bond Predictor**: This predictor takes an additional atom coordinate as input, which is processed through GNN layers followed by additional GVP and MLP layers. It produces logits for all element types and bond types for each existing atom, using a new type of self-attention module to enhance the performance.

C.2 Sampling Method

The sampling method of Pocket2Mol implemented for the original paper involves a repetition of atom sampling, threshold-based filtering, ranking, and pool addition until the number of generated molecules reach a certain target.

C.2.1 Atom Sampling

A set of candidate atoms are generated from each data in the queue through the following steps:

- **Focal Atom Sampling**:
  - **For the initial atom**: The frontier probability is calculated for each protein atom by the frontier module.
  - **For subsequent sampling**: The frontier probability is calculated for each atom in the partially generated molecule by the frontier module.
  - All atoms with the frontier probability higher than 0.5 are designated as frontiers, which then serve as candidate focal atoms.
  - A finish signal is returned if no frontier atom was identified.
- **Position Sampling**:
  - For each one of the n_{focal} candidate focal atoms, all $n_{\text{component}}$ mean positions of the Gaussian mixture distribution predicted by the atom coordinate module become the candidate atom coordinates.
- **Element and Bond Type Sampling**:
  - For each pair of focal atom and atom coordinate candidates, $n_{\text{atom\_samples}}$ atom type candidates are sampled by the atom type module.
  - **For subsequent sampling**: Along with the atom type sampling, $n_{\text{bond\_samples}}$ combinations of bond types between all existing molecule atoms are sampled by the bond type module to form the bond type candidates.
  - Duplicates are eliminated from the combinations of element and bond type candidates. For the initial atom, only duplicated atom type candidates are removed, as bond types do not appear yet.

To summarize, at most $n_{\text{focal}} \times n_{\text{component}} \times n_{\text{atom\_samples}} \times n_{\text{bond\_samples}}$ atom configurations are sampled.

C.2.2 Threshold-based Filtering

During the sampling of each component of the atom configuration at the previous stage, the probability (or the probability density function) that the corresponding module samples the value is recorded. At this stage, only the configurations whose components have probabilities higher than the designated thresholds are retained.

C.2.3 Ranking and Pool Addition

The candidate configurations are ranked according to the sum of log probabilities, and only the top candidates (this number is referred to as "beam size") proceed to the next iteration. Each proceeding configuration results in a unique addition of an atom to the molecule being generated, and the resulting extended molecule is entered into the queue.

D Details of reinforcement learning

D.1 Optimization Algorithm

Proximal Policy Optimization (PPO), employed by our reinforcement learning method, is a policy gradient algorithm renowned for its stability and simplicity (Schulman et al. (2017)). It achieves this stability through gradual, incremental updates to the policy, ensuring consistent and controlled improvements.

The core mechanism of PPO involves the use of a clipping loss function. This function constrains the policy's updates, keeping the gradient changes within a specific range relative to the reference model.

Let $r_t(\theta)$ be the probability ratio, defined as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. The loss in PPO is computed as follows.

$$L^{CLIP}(\theta) = \hat{\mathbf{E}}_t\big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\big] \tag{3}$$

D.2 Failure types of episodes

As explained in 3.2, an episode rollout can fail in three ways. Type I failure occurs when either (1) the resulting graph cannot be parsed into a valid RDKit Mol object or (2) the computation of one of the reward terms terminates with an error. Type II failure occurs when the StopAction is not returned until a certain number (default: 50) of atoms is generated. Type III failure occurs when the sampled atom action either (1) has no bond or (2) has a bond that makes an existing atom's total bond count exceed its typical valence range. The thresholds for the "typical valence range" are set to C: 4, N: 3, O: 2, F: 1, P: 5, S: 6, and Cl: 1. The rationale behind this failure type is that such circumstances would ultimately result in either type I or type II failure, so it would be beneficial to return the failure as soon as possible for a better reward signal.

D.3 Training curriculum

We train our model in two stages.

In the first stage, we focus on decreasing the occurrence of episode failures and fixing the stereochemical problems while not falling into shortcut solutions that achieve the goals. To do so, we use the stereochemical measurements (those in Table 2) and the QED score as the reward terms. A low QED score would be given to shortcut solutions to the problem of generating chemically valid molecules with valid conformers, such as those without any ring system. Therefore, including it in the reward would prevent the learning process from converging to such undesirable local optima. During the first stage of training, we freeze some parts of the model in order not to lose the information learned from the supervised training of Pocket2Mol. These frozen parts are the embedding layers, the encoding layers, and the GNN layers used for the atom type and the bond type modules.

In the second stage, our focus shifts to enhancing the binding affinity and drug-likeness while preserving the chemical and structural validity attained in the first stage. To do so, we additionally use the Vina score measured without Vina optimization as a reward. Also, we decrease the focus on the QED score and stereochemical validity by setting the reward scaling factor α_i to 0.5 for the QED score and 0.2 for the fractions of stereochemical problems. Note that the reward statistics $\text{mean}(R_i)$ and $\text{std}(R_i)$ in equation 1 are calculated based on the model checkpoint obtained from the first stage of training, which is used to initialize the second stage. The parts of the model frozen in the first stage are frozen in the second stage as well. In addition, we freeze the parts of the atom position module that determine the means (μ) of the Gaussian mixture distribution to preserve the stereochemical constraints learned from the first stage more reliably.

D.4 Model selection

At each stage of the curriculum, we conduct multiple parallel runs of training and select the model with the highest average reward on the validation set as the final model.

D.5 The critic model

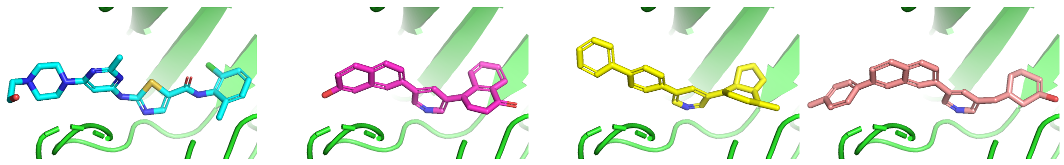
While the actor model has the same architecture as the original Pocket2Mol model, the critic model is constructed via a slight modification from the actor model to produce scalar outputs. Specifically, more GVP layers were stacked after the encoder, followed by an MLP layer, a mean-reduction operation over the nodes, and another MLP layer.

At each stage of training, we initialize the parameters in the parts of the critic model shared with the actor model using the same values that we use to initialize the actor model. In the second stage of training, we train only the critic model without training the actor model, for a certain number of epochs. This choice reflects our judgment that the objectives of the second stage are harder to achieve, requiring several epochs for the critic model to become effective.

D.6 Computational resources

We used three A100 GPUs (80GB RAM each), supporting three parallel runs each, totaling nine parallel runs across both stages of training. The first and second stages required approximately 24 and 46 hours, respectively.

E Example molecules



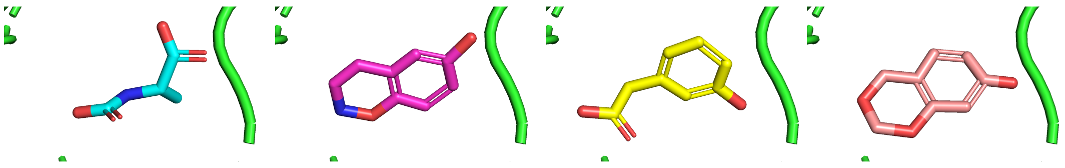
(a) reference molecule
- Affinity: -10.0
- QED: 0.47
- SA: 0.82

(b) molecule 1 (22.sdf)
- Affinity: -11.1
- QED: 0.49
- SA: 0.79

(c) molecule 2 (23.sdf)
- Affinity: -11.0
- QED: 0.47
- SA: 0.74

(d) molecule 3 (96.sdf)
- Affinity: -10.9
- QED: 0.38
- SA: 0.78

Figure 2: The reference molecule of 4XLI and molecules generated from the protein by our model



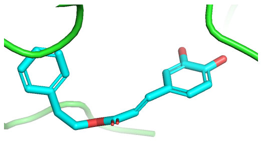
(a) reference molecule
- Affinity: -4.6
- QED: 0.48
- SA: 0.84

(b) molecule 1 (58.sdf)
- Affinity: -5.8
- QED: 0.58
- SA: 0.79

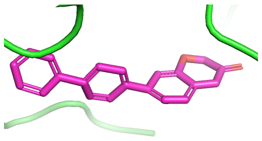
(c) molecule 2 (61.sdf)
- Affinity: -5.8
- QED: 0.66
- SA: 0.93

(d) molecule 3 (71.sdf)
- Affinity: -5.7
- QED: 0.61
- SA: 0.85

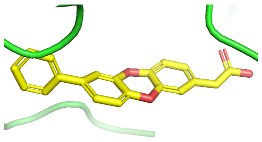
Figure 3: The reference molecule of 3L3N and molecules generated from the protein by our model


(a) reference molecule

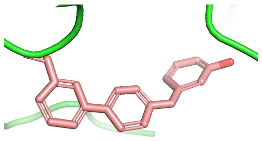
- Affinity: -7.7
- QED: 0.50
- SA: 0.89


(b) molecule 1 (75.sdf)

- Affinity: -10.3
- QED: 0.70
- SA: 0.89


(c) molecule 2 (77.sdf)

- Affinity: -10.2
- QED: 0.58
- SA: 0.88


(d) molecule 3 (60.sdf)

- Affinity: -9.8
- QED: 0.69
- SA: 0.90

Figure 4: The reference molecule of 5LIU and molecules generated from the protein by our model

Examples of generated molecules are illustrated in Figures 2, 3, and 4. The target proteins and the reference molecules are from the PDB entries 4XLI, 3L3N, and 5LIU, which were the first to third ones in our test set. The provided generated molecules are those with the highest binding affinity among the molecules used to form Table 1, excluding those with any stereochemical problems listed in Table 2. It can be observed that (1) the generated molecules have much higher binding affinity than the reference, (2) the generated molecules have QED and SA not worse than those of the reference on average, (3) the spatial spans of the generated molecules are similar to that of the reference molecule, and (4) the generated molecules are not overly simplistic. The last two points indicate that the performance metrics in Table 1 and Table 2 were achieved without resorting to shortcut solutions.

F Failed attempts

As well as the promising results, we would like to share some failed attempts, with the hope of providing useful insights to the readers.

F.1 Using shortcut-prone rewards

Using reward terms with a shortcut solution often resulted in models that generated faulty molecules. For example, we initially tried the change in energy during Universal Force Field optimization, normalized by a power of the molecule size, as a reward term, hoping that it would serve as a stereochemistry indicator. However, we observed that the training process often led the model to generate extremely small or big molecules, depending on the exponent for normalization. We believe that this was a consequence of the existence of shortcut solutions to the reward term.

F.2 Using highly parameterized rewards

Using highly parameterized and empirically tuned reward terms may lead to a *reward misalignment* problem. For example, we tried using the Molskill (Choung et al. (2023)) score, a scoring function based on deep learning tuned to predict the preference of human experts, as a measure of drug-likeness and synthetic accessibility. The training seemed to be flawless according to the validation metrics. The Molskill score was successfully optimized, and there was no obvious problem, such as the size collapse described in the preceding subsection. However, a careful examination revealed that the generated molecules were not of typical shape. For example, most of them had no 5- or 6- rings. We believe that this discrepancy between the Molskill score and the seeming sanity of molecules stems from the highly parameterized and empirically tuned nature of Molskill. Rewards of this nature may not work as expected for molecules generated during reinforcement learning if there is a *distributional shift* from the molecules used to tune the reward.

There might be two potential solutions to this problem. Firstly, the reward can be continuously fine-tuned during reinforcement learning based on newly obtained human annotations on the molecules generated by the actor model. This approach of *iterative fine-tuning* was adopted by a successful RLHF implementation (Touvron et al. (2023)). Secondly, additional devices can be used to confine the generated molecules to the region of feature space where the reward maintains reasonable accuracy. For example, a *feature matching* reward similar to the one used for GAN (Salimans et al. (2016)) can be used along with the scoring reward itself.

F.3 KL DIVERGENCE PENALTY

In RLHF implementations (Ouyang et al. (2022), Ziegler et al. (2020), Touvron et al. (2023)), the technique of penalizing the model with KL divergence between the distributions specified by the current and pre-trained actor models was used to prevent the model from forgetting the information learned from the previous stages of training. We tried the same approach when experimenting with our method on a task similar to molecule generation, with the same forms of inputs and outputs as molecule generation, but have not found notable improvements in performance. Currently, we lack specific insights as to why this occurred.