Zero-Shot 3D Drug Design by Sketching and Generating

Siyu Long[*1], **Yi Zhou**[2], **Xinyu Dai**[1], **Hao Zhou**[3]
[1]National Key Laboratory for Novel Software Technology, Nanjing University
[2]ByteDance AI Lab
[3]Institute for AI Industry Research (AIR), Tsinghua University
longsy@smail.nju.edu.cn, zhouyi.naive@bytedance.com
daixinyu@nju.edu.cn, zhouhao@air.tsinghua.edu.cn

Abstract

Drug design is a crucial step in the drug discovery cycle. Recently, various deep learning-based methods design drugs by generating novel molecules from scratch, avoiding traversing large-scale drug libraries. However, they depend on scarce experimental data or time-consuming docking simulation, leading to overfitting issues with limited training data and slow generation speed. In this study, we propose the zero-shot drug design method DESERT (**D**rug d**E**sign by **S**k**E**tching and gene**R**a**T**ing). Specifically, DESERT splits the design process into two stages: sketching and generating, and bridges them with the molecular shape. The two-stage fashion enables our method to utilize the large-scale molecular database to reduce the need for experimental data and docking simulation. Experiments show that DESERT achieves a new state-of-the-art at a fast speed.[1]

1 Introduction

Drug design is a crucial step in the drug discovery cycle, which is the inventive process of finding new drugs based on a biological target (usually a protein pocket) [1, 2, 3]. However, seeking appropriate drugs for a particular target is quite challenging due to the enormous space of drug candidates (almost 10^{33}). Traditional drug design approaches usually employ virtual screening [4, 5, 6] and molecular dynamics [7, 8] to traverse in a large scaled drug library, which is time-consuming and could not produce novel drug candidates. Recently, a line of work proposes to realize drug design by generating drug molecules from scratch using deep generative models [9, 2, 10], which is quite promising due to the fast speed and the ability of *de-novo* drug design.

Most of current drug generation model are developed upon 1D (SMILES) [11, 12, 13] or 2D (molecular graph) [14, 15, 16, 17, 18, 19] molecular structures, which heavily rely on expensive experimental data for supervised training while ignoring the 3D interaction information between the drug and the pocket. In a word, they attempt to find a molecule that maximizes the score given by a bio-activity predictor trained on experimental data. They employ different optimization methods, such as Generative Adversarial Network (GAN) [20], Bayesian Optimization (BO) [21, 22, 9], Reinforcement Learning (RL) [23, 24, 20, 25, 26, 27], Evolutionary and Genetic Algorithms (EA/GA) [28, 29, 30, 31, 32], and Markov Chain Monte Carlo (MCMC) [33, 34], in the molecular space for obtaining desired drug molecule under certain constraints. However, we argue that a purely data-driven approach of drug design is practically limited, since in most cases, the quantity of experimental drug-pocket pairs hardly enables supervised training of *de-novo* drug design. Generally, most protein pockets lack bio-activity data, and learning on noisy and deficient data may lead to severe overfitting problems.

[1]Code is available at https://github.com/longlongman/DESERT.
∗ Work was done when Siyu Long was a research intern at Bytedance AI Lab.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

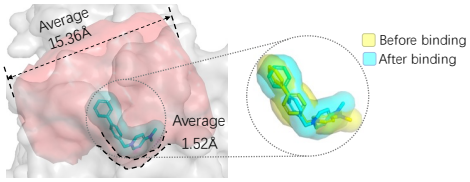


Figure 1: When a drug is binding to a pocket, its shape does not change too much (with an RMSD less than 1.391Å) and is complementary to the pocket.

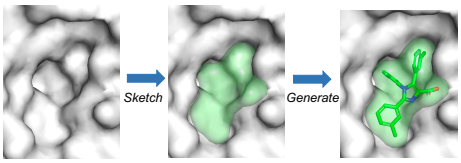


Figure 2: DESERT splits the whole drug design process into two stages: sketching the shape and generating the molecules.

Recently, several drug design models have been proposed to directly generate drug molecules in the 3D space, the realistic space of drug-target interaction. Generating in such space is very promising for the potential of leveraging some prior knowledge (e.g., physical knowledge) instead of entirely counting on data-driven methodology. Specifically, Masuda et al. [35] and Luo et al. [2] propose efficient 3D generative models to learn the atom density conditioned on protein pockets, with GAN and auto-regressive models, respectively. Nevertheless experimental data are still obligatory in their models for achieving satisfactory results. Even more noteworthy is GEKO [10], which combines the ideas of 3D generation and physical simulation to obtain state-of-the-art drug design performance without the help of large-scale experimental data. Intuitively, GEKO performs geometric editing in the 3D molecule space guided by docking simulation (physical knowledge) [36, 37]. However, GEKO may suffer from two concerns: a) the frequent invocation of docking is very time consuming, which significantly slows down the speed of the drug design model [38, 39]. b) docking may not always be accurate enough, especially in some complex settings [40, 41]. In such a case, being heavily dependent on the docking accuracy could hurt the generalization of the proposed drug design model.

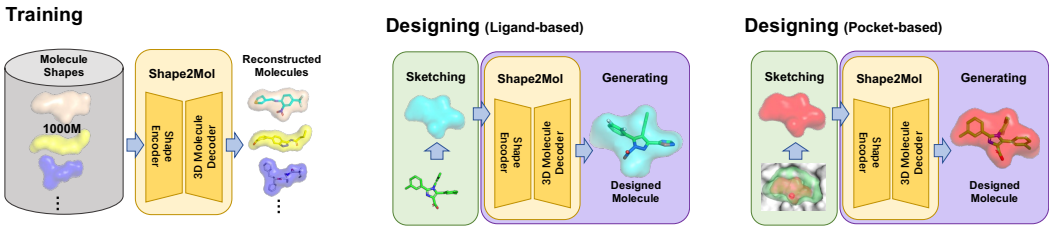
In this paper, we propose the zero-shot approach DESERT, namely **D**rug d**E**sign by **S**k**E**tching and gene**R**a**T**ing. Motivated by the idea of *structure determines properties* [42, 43, 44, 45], DESERT is built on the assumption that molecular shape determines bio-activity between drug molecules and its target pocket. In other words, we suppose that a drug candidate would have satisfactory bio-activity to a target pocket if their shapes are complementary (see Figure 1).

With such prior, as shown in Figure 2, DESERT splits the whole drug design process into two stage: sketching and generating, which employs the *molecular shape* as the bridge of the two stages. Such splitting makes DESERT enjoy two advantages: a) DESERT does not heavily rely on docking simulation, which only optionally uses docking for post-process and thus avoids the aforementioned disadvantages of GEKO. b) DESERT abandons the expensive experimental data. Specifically, in the sketching stage, we only need to sample some reasonable shapes complementary to the target pocket. In the generating stage, DESERT proposes to employ a generative pre-trained model (from shape to concrete molecular) to fill the shape obtained in the last sketching stage. Notably, the generative pre-trained model is only trained on the ZINC database, which contains 1000M pairs of molecules and their corresponding shapes. This process does not rely on experimental data, making DESERT work in a zero-shot fashion. [2]

Note that DESERT is not baseless. Besides the idea that structure determines properties (an important concept of Structural Biochemistry), we also have preliminary results to verify our assumption. Figure 1 shows that after binding to a pocket, the root-mean-square deviation (RMSD) is not too shabby (1.391Å) compared with molecular conformation generation methods such as CGCF [46] (1.248Å) and ETKDG [47] (1.042Å). The results suggest that the molecular shape is stable when molecules bind to proteins. We also show in Figure 1 that a ligand often attaches tight to a pocket, which means their shape is complementary.

In summary, our contributions are three-fold: (1) We propose DESERT, a generative method for *de-novo* 3D shape-based drug design in the zero-shot setting. (2) DESERT trains a pre-trained model

[2]The generation stage of DESERT can also be equipped with chemistry priors, we conduct some experiments in Appendix 2.2.



(a) Pre-training with massive unbound molecules

(b) DESERT with ligand-based sketching

(c) DESERT with pocket-based sketching

Figure 3: Overview of DESERT. In Figure (a), we use massive unbound molecules[2] to train the SHAPE2MOL, which includes two sketching variants. In Figure (b) and (c), we use the SHAPE2MOL to generate 3D molecules to fill in the given shapes. In Figure (b) where existing drugs are available, we treat their shapes as desired molecule shapes (ligand-based). In Figure (c) where only protein pockets are given, we heuristically sample reasonable molecular shapes from the pockets (pocket-based).

from massive unbound molecules [3], eliminating the constraints of labeled data. (3) DESERT achieves a new state-of-the-art result (the docking score improves 0.79kcal/mol and 2.93kcal/mol over the best supervised method on two datasets) at a fast speed (about 20 times faster than GEKO[4]).

2 Proposed Method: DESERT

In this section, we describe the proposed method in detail. Inspired by the two aforementioned preliminary studies (see Figure 1 and Figure 2), DESERT designs drugs for proteins in a two-stage fashion and employs the shape as a bridge since previous work has shown the feasibility of designing drug by molecular shape [49, 50, 51, 52, 53].

Specifically, DESERT designs drugs by first sampling appropriate shapes complementary to the target pocket and then mapping the shapes to specific molecules. In Section 2.1, we first introduce the overall picture of how DESERT works in a zero-shot setting. Then we pose two challenges: how to sketch the reasonable molecular shapes and how to generate corresponding molecules based on the shapes. We put forward solutions in Section 2.2 and Section 2.3, respectively.

2.1 Zero-Shot Pipeline

In this section, we focus on how DESERT designs drugs in the zero-shot setting. Briefly speaking, DESERT produces molecules in a two-stage fashion: sampling the shape of the desired drug first (sketching) and generating molecules conditioned on the resulting shape (generating).

Zero-Shot Sketching There are mainly two cases when DESERT needs to sample molecular shapes. In the zero-shot case where no reference protein ligands are available, DESERT samples reasonable shapes from protein pockets (see Figure 3c) based on biological observations. Besides, DESERT can also reuse the shape of a ligand to design a novel one (see Figure 3b). Details are listed in Section 2.2.

Zero-Shot Generating DESERT generates molecules through a pre-trained generative model, namely SHAPE2MOL, which can convert a given shape into diverse molecules (see Figure 3a). In this procedure, we utilize massive unbound molecules to train the model. Thus no information about proteins is needed. Details of this model are presented in Section 2.3.

2.2 Sketching Molecular Shapes

The sketching stage is responsible for deciding what desired molecules look like. In this section, we show how we design two heuristic methods to sketch the shapes of desired molecules.

[3]For clarity, the following terms will be used throughout this study: "bound drug/molecule" (or "unbound drug/molecule") refers to the drug/molecule that is bound (or unbound) to proteins [48].

[4]We calculate the speed by measuring the time from given a pocket to getting 100 molecules.

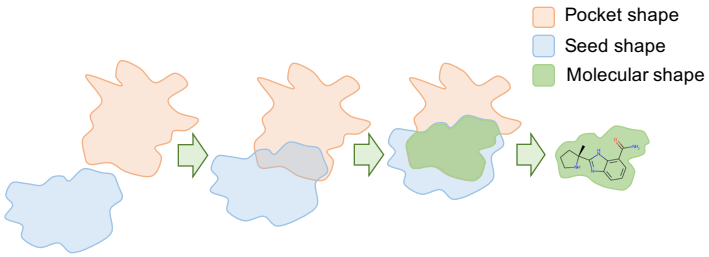


Figure 4: A 2D illustration of sampling molecular shapes from pockets. The 2D molecule in the figure represents a potential drug that fits the molecular shape.

There are mainly two cases when we sketch a molecule shape based on whether the ligand is provided. When the ligand is available, the sketching process can be trivial since molecules with similar shapes have similar properties. It is reasonable to directly use the ligand's shape as the shape of the desired molecule, which we call **Ligand-based Sketching**. If the ligand of the pocket is unavailable, the challenge is how to obtain a shape that has a high potential to bind to a pocket. [5] We name this **Pocket-based Sketching**, and our main idea is to sample a region with an appropriate size complementary to the surface of the pocket.

Our idea for Pocket-based Sketching is based on two main observations:

1. Ligands mainly lie in the area close to the pocket surface. Figure 1 shows that the shape of satisfactory ligand is tightly complementary to the pocket.
2. Pockets are usually much larger than ligands, suggesting that directly utilizing the shape of pockets to design molecules is inappropriate.

To this end, we present an algorithm (see the Appendix 1.1) to obtain the desired molecule shape, which is of the appropriate size and complementary to the pocket surface. We achieve this goal by finding another shape (namely seed shape) that intersects with the pocket, where the intersection has a similar size to a molecule. We also show a 2D illustration in Figure 4.

2.3 Generating 3D Molecules by SHAPE2MOL

In this section, we introduce SHAPE2MOL, an encoder-decoder network mapping a shape to diverse and high-quality 3D molecules. There are plenty of unbound molecule data, e.g., 1000M molecules in the ZINC database, making it possible to learn a large-scale pre-trained generative model from shape to molecule.

Concretely, we formulate the problem as an image-to-sequence generation, where the shape is voxelized as a 3D image (see 2.3.1), and the 3D molecule is converted to be a sequence (see 2.3.2). Our generative approach is capable of modeling any complicated molecule structure and the linearization makes large-scale pre-training easier to implement.

2.3.1 Encoder: Voxelized Shape

The shape encoder is a 3D extension of the ViT [56], where we use 3D patches instead of 2D patches in the original ViT. Let $\mathcal{A}$ denote the set of all atoms, a molecule m can be constructed as a collection of atoms and their corresponding coordinates:

$$m = \{(a, c) | a \in \mathcal{A}, c \in \mathbb{R}^3\}$$

Given a molecule m, we transform its shape into a 3D image with a voxelization function $v_m : \mathbb{Z}^3 \to \{0, 1\}$:

$$v_m(x, y, z) = \begin{cases} 1 & \exists (a, c) \in m, \|(x, y, z) - c\|_2 \leq r(a) + \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where r denotes the Van der Waals radii [57], ϵ is a perturbed noise which helps prevent overfitting [58].

[5] The pocket can be generated by CAVITY [54] or f-pocket [55]. In this study, all protein pockets are generated by CAVITY.

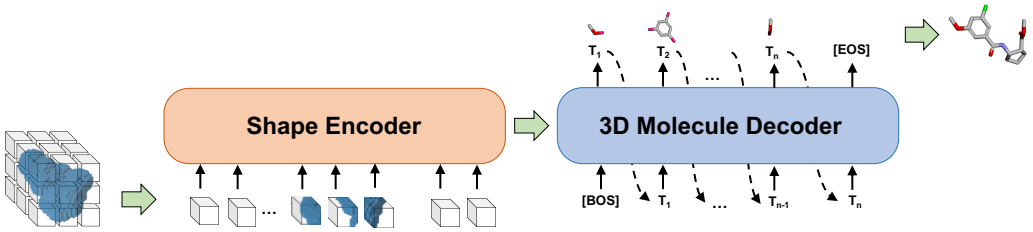


Figure 5: The architecture of SHAPE2MOL.

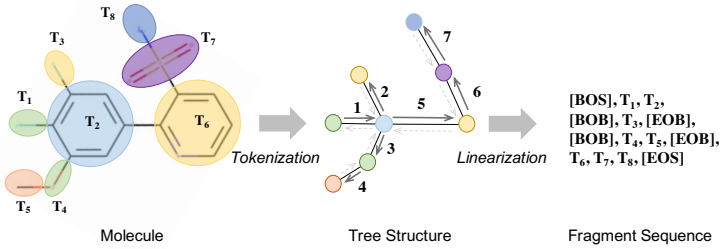


Figure 6: The process of converting a molecule into a sequence.

2.3.2 Decoder: Linearized Molecule

The molecule decoder in our model is similar to the Transformer decoder in machine translation [59]. The main difference is that the decoding object here is a 3D molecule instead of a 1D sequence. To address this, we propose a 3D molecule decoder, which handles a 3D molecule as a sequence of tuples. The sequence object eases the implementation of a pre-trained model. To obtain the object, we first cut a molecule into pieces, then convert it to a sequence.

Tokenization We cut a molecule into pieces so that the generative process can be easily factorized. Our principles are three folds: (i) preserving the functional groups since they are vital for determining molecule properties [60], (ii) avoiding too large size of the vocabulary to ease the pre-training process [61], (iii) no circles exist in the segmented molecules since a tree structure is simpler to handle than a graph. Our method is simple yet efficient: first tokenizing the molecules with BRCIS [62] and then cutting all single bonds attached to a ring. More details about some pilot experiments can be found in Appendix 1.2.

Linearization For the network output, we propose to utilize a linearized sequence to represent the target molecule graph, which is not only convenient for training, but also has the strong power to represent any complicated tree structure. We first select an fragment whose degree is 1 as the root of the tree (e.g., `T1`, `T3`, `T5`, and `T8` in Figure 6). Then we traverse the tree in the depth-first-traverse style[63]. Whenever we enter or leave a branch, we will add two special symbols `[BOB]` and `[EOB]` (beginning/ending of a branch), respectively.

More particularly, we use a tuple (C, P, R) to represent a fragment F, where $C = \mathbf{1}_F$ is an indicator function [64] denoting its index in the vocabulary, $P \in \mathbb{R}^3$ is the translation vector [65], $R \in \mathbb{R}^4$ is the rotation quaternion [66]. In order to stabilize the training process, we further discretize the continuous variable P and R into P^c and R^c, respectively. Taking the translation vector P as an example, we convert it into a binary vector P^c, which satisfies:

$$P^c[i] = \begin{cases} 1 & \lfloor \frac{L}{b} i \rfloor \leq P < \lceil \frac{L}{b} i \rceil \\ 0 & \text{otherwise} \end{cases}$$

where L is the max translation length, b is the bin size.

2.4 Training & Decoding

Training Given the output probabilities of the model $(\hat{C}_i, \hat{P}_i^c, \hat{R}_i^c)$, which denote the probability of a fragment, a discreterized translation vector, and a discreterized rotation vector, respectively. We calculate the corresponding cross-entropy loss and use their sum as the final loss function.

$$\mathcal{L} = -\sum_{i=1}^{n} \left\{ C_i \log \hat{C}_i + P_i^c \log \hat{P}_i^c + R_i^c \log \hat{R}_i^c \right\} \tag{1}$$

where n is the length of fragment sequence.

We use 100M unbound molecules sampled from the lead-like subset of ZINC as the training data. The Transformer's dimension is 1024, and both encoder and decoder have 12 stacked Transformer layers. When training SHAPE2MOL, we set the dropout rate as 0.1, batch size 2048, train step 300K and use AdamW [67] with learning rate 5e-4, weight decay 1e-2, and warmup step 4000 as the optimizer. The model is developed by ParaGen [68] [6] and trained on 32 Telsa V100 GPU cards for 2 weeks. Following [35], we also randomly rotate and translate the input shape for invariance.

Decoding We design a decoding strategy to provide diverse and high-quality candidate molecules for a given protein. To achieve diversity, we employ the sampling method Nucleus [69] to generate multiple fragment sequences. Then we convert the sequences back to molecules with a greedy algorithm (see Appendix 1.2), which connects the fragments by greedily enumerating the nearest pair of breakpoints. [7] Finally, we do some post-processing operation to further improve the diversity and quality. We remove the duplicate molecules and leverage the docking simulation to drop molecules that do not pass the affinity threshold.

When testing SHAPE2MOL, we set the threshold of Nucleus sampling to 0.95. For each protein pocket, we sketch 200 shapes. For each shape, we generate 1000 molecules. More details of SHAPE2MOL can be found in the Appendix 1.3.

3 Results and Discussions

3.1 Experiments

Data We evaluate the performance of our method on drug design by using a total of 12 proteins (PDB IDs: 1FKG, 2RD6, 3H7W, 3VRJ, 4CG9, 4OQ3, 4PS7, 5E19, 5MKU, 3FI2, 4J71), which is a combination of the test data used in Masuda et al. [35] and Jin et al. [23]. Among the proteins, only JNK3 and GSK3β (PDB IDs: 3FI2, 4J71) have sufficient labeled data to train well-performing bioactivity predictors. We denote the set of these two proteins as Set B and the rest as Set A. Because 1D/2D methods need bioactivity predictors to design drugs for given proteins, we only evaluate them on Set B while evaluating 3D-based methods on both sets.

Baselines We compare DESERT with some baselines for drug design. Based on the resource needed for designing drugs, we further divide these baselines into three groups: **Guided** methods need docking simulation or extra bioactivity predictors to provide supervision signals. **Supervised** methods rely on labeled data to train their models. **Retrieved** methods directly search the database for desired molecules.

Evaluation Following Yang et al. [10], we evaluate the performance of methods from two aspects. (1) the molecular space covered by designed results. (2) the capacity to provide highly active molecules. As a high-quality molecular space should contain diverse, novel molecules with pharmaceutical potential, we use five metrics to evaluate the molecular space: Uniqueness (**Uniq**), Novelty (**Nov**), Diversity (**Div**), Success rate (**Succ**), and Product (**Prod**). To evaluate the ability to provide highly active molecules, we compare the distributions of $\mathrm{Vina}_{\mathrm{score}}$ and use Median Vina Score (**Median**) to quantify the distribution. More details about these metrics can be found in Appendix 2.1.

Detail of Generation We describe how different methods generate molecules for comparison. For each protein, every method needs to generate 100 molecules for comparison. For 3D methods, we

[6] https://github.com/bytedance/ParaGen

[7] Whenever we cut a chemical bond in tokenizing, we mark the atoms of the chemical bond as breakpoints.

Table 1: Performance comparison among drug design methods. ↑ indicates higher is better. ↓ indicates lower is better.

Targets	Method		Uniq (%)↑	Succ (%)↑	Nov (%)↑	Div ↑	Prod ↑	Median (kcal/mol)↓
Set A	Guided	GEKO [10]	100.0	55.7	100.0	0.912	0.51	-9.58
	Supervised	liGAN [35]	100.0	0.4	100.0	0.924	0.00	-5.84
		3D SBDD [2]	69.7	13.6	98.9	0.839	0.08	-8.83
	Retrieved	SCREEN (1K)	100.0	25.6	100.0	0.892	0.23	-7.46
		SCREEN (200K)	100.0	64.0	100.0	0.889	**0.57**	-8.66
	Ours	DESERT-LIGAND	100.0	65.3	87.0	0.786	0.41	-8.89
		DESERT-POCKET	100.0	61.1	100.0	0.908	**0.57**	**-9.62**
Set B	Guided	JT-VAE [9]	100.0	13.0	100.0	0.907	0.12	-8.35
		RationaleRL [23]	100.0	27.0	35.0	0.884	0.08	-7.75
		GA + D [30]	39.0	24.0	87.0	0.852	0.06	-7.22
		GraphAF [26]	97.0	0.5	100.0	0.946	0.00	-4.22
		MolDQN [27]	76.5	0.0	100.0	0.742	0.00	-5.52
		MolEvol [70]	99.5	40.5	63.5	0.742	0.17	-8.19
		MARS [33]	86.0	31.5	93.0	0.805	0.22	-7.68
		GEKO [10]	100.0	57.0	100.0	0.910	0.52	-9.19
	Supervised	liGAN [35]	99.8	0.2	100.0	0.923	0.00	-5.34
		3D SBDD [2]	99.9	5.2	100.0	0.853	0.05	-6.39
	Retrieved	SCREEN (1K)	100.0	3.0	100.0	0.891	0.03	-6.94
		SCREEN (200K)	100.0	32.0	100.0	0.882	0.28	-7.95
	Ours	DESERT-LIGAND	100.0	18.0	100.0	0.913	0.17	-7.34
		DESERT-POCKET	100.0	61.0	100.0	0.907	**0.55**	**-9.32**

further use the local minimization module in Vina [36] to optimize the generated structures. For 1D/2D methods, we first use RDKit to generate the 3D conformer of their results. For method SCREEN, we sample 1K and 200K molecules from ZINC and return 100 molecules with the highest $\mathrm{Vina}_{\mathrm{score}}$ as the generated results.

3.2 Main Results

Table 1 shows the results of DESERT and baselines. All values are averaged over target proteins. Our main findings are listed as follows:

I.The zero-shot DESERT *achieves the SOTA result at a fast speed.* DESERT's performance is strong compared with GEKO, an MCMC-based model with a huge sampling space. Compared with GEKO, which works in a trial-and-error way, DESERT makes a more clever choice by pruning the space with its biological knowledge regarding the shape and quickly finds a good solution with limited hints from a teacher. We compare the generation speed in Figure 7.

II.The shape helps DESERT *produce high quality molecules.* The molecular space of SCREEN is the ZINC database, while that of DESERT is generated by the pre-trained model SHAPE2MOL, which is aware of the pocket's shape. According to the table, both DESERT-LIGAND and DESERT-POCKET show observably better performance than their counterparts, i.e., SCREEN (1K) and SCREEN (200K).

III. More comprehensive exploration of protein pockets benefits performance. Instead of using the molecular shapes of reference ligands as input, DESERT-POCKET comprehensively explores the protein pockets by sampling multiple molecular shapes from them. Therefore, it has the potential to obtain diverse, high-quality molecules that bind to protein pockets in different regions. While DESERT-LIGAND only considers one region, i.e., the region that ligands lie in, limiting the exploration of protein pockets. We show a case in Figure 8.

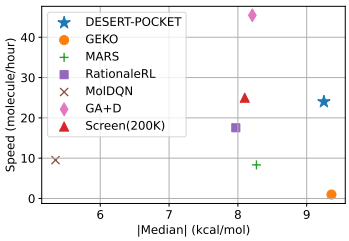


Figure 7: The absolute value of the median vina score and the speed of different methods on target protein 3FI2.

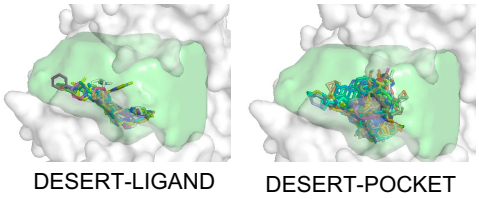


Figure 8: Molecules generated by DESERT on protein 4CG9.

Table 2: Comparison of shape faithfulness and structure rationality. "Random" is the Shape Tanimoto between two random molecules, "Real" is the Free Energy of reference ligands.

Method	Shape Tanimoto	Free Energy (kcal/mol)
Random	0.325	/
Real	/	167.28
liGAN (Ligand)	0.869	289.55
DESERT-LIGAND	0.875	188.54

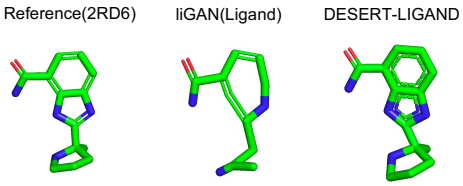


Figure 9: A cases from liGAN and DESERT.

IV. Unsupervised methods have larger potential than the supervised counterparts. As labeled data is inadequate, e.g., scPDB [8] only has 16,034 entries, the supervised methods easily collapse to the main molecule pattern in their dataset. When generating molecules with them, we often get the same molecules, e.g., 3D SBDD can only generate 16 unique molecules for protein 4OQ3. DESERT utilizes massive unbound molecules, which leads to the learned space being denser. Combing with an appropriate sampling method, it can generate diverse molecules.

3.3 Comparison with Related Shape-based Models

In this section, we study the shape faithfulness and structure rationality compared with previous shape-based models. In Table 2, we use Shape Tanimoto [9] to evaluate the faithfulness and use Free Energy to quantify the rationality [74]. We compares DESERT-LIGAND with liGAN (Ligand), a variant of liGAN [35] which utilizes the existing ligands. Although liGAN (Ligand) achieves high performance on Shape Tanimoto, its atom-based decoding strategy does not guarantee the correct relative position between atoms. Therefore, liGAN shows higher Free Energy, indicating that unrealistic structures may appear. Figure 9 shows a case where liGAN produces an ill ring structure.

3.4 Ablation Study of Generating

In this section, we evaluate several designs of the generating stage in our DESERT method, which relate to the pre-trained model and decoding strategy. All results are based on an extra test set from ZINC and a smaller version of SHAPE2MOL. [10]

Pre-training Configuration We evaluate the model quality on different pre-training configurations (mainly focusing on the size of the model and training data). The results in Figure 10 show: (1) Larger model achieves better performance. Devlin et al. [75] observes similar phenomenons in natural language modeling. (2) Performance saturation occurs when the dataset is of moderate size. As the

[8] scPDB is a high-quality labeled dataset for 3D drug design.

[9] Shape Tanimoto [71, 72, 73] measures the similarity between the input and generated molecules. $\text{Shape Tanimoto}(A, B) = \frac{A \cap B}{A \cup B}$, where A and B are two molecular shapes.

[10] The extra test set contains 10K molecules. The smaller SHAPE2MOL has 512 model dimension and 6 layers of encoder and decoder. We use a greedy decoding strategy and remove post-processing.

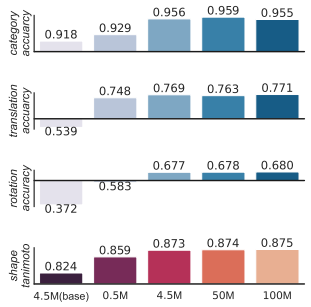


Figure 10: Comparison of different pre-training configurations.

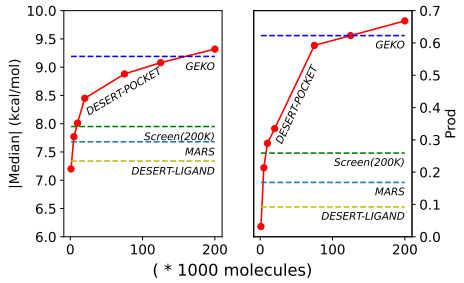


Figure 11: Comparison of sampling space size.

Table 3: Performance comparison on more protein targets. Because the original test data only contains incomplete protein pockets, we recover the complete pockets by aligning the incomplete pockets to the structures from PDB database. We managed to recover 55 pockets and apply DESERT-POCKET to them. For 3D SBDD, we use the released code and apply the same post-processing used in DESERT to it.

Metric Med.	3D SBDD w/o post-processing	3D SBDD w post-processing	DESERT-POCKET w/o post-processing	DESERT-POCKET w post-processing
Vina Score (kcal/mol)	-6.069	-7.584	-6.148	-9.410
QED	0.522	0.501	0.614	0.549
SA	0.672	0.623	0.612	0.616
Diversity	0.873	0.826	0.926	0.908

map from radii to atoms is easy to learn, the model can capture it with a moderate dataset. Liu et al. [76] reports a similar result that a large dataset does not necessarily lead to better quality.

More Ablation We also do ablation study about some model variants (including *discretization* and *robust training*) and *decoding strategies*. We further study *chemical information driven design* and *atom-based pre-training*. We refer the reader for more details to the Appendix 2.2.

3.5 Ablation Study of Sketching

In this section, we study the sketching stage in our DESERT method, which includes the effect of sampling space size and seed shape on the method's performance. All the results are based on the same DESERT-POCKET method in Section 3.2 and are calculated on Set B.

Sampling Space Size In Figure 11, we evaluate the performance of DESERT with respect to sampling space size, i.e., the total number of generated molecules before post-processing. The results show: (1) Increasing sampling space size leads to better performance. With a larger sampling space, DESERT finds more molecular shapes complementary to pockets, leading to a performance rise. (2) The shape can effectively prune the sampling space for screening. Instead of directly searching molecular space, DESERT achieves a similar performance by pruning the space from 200K to 10K with molecular shapes.

More Ablation We also do ablation study about the usage of different *seed shape*. We refer the reader for more details to the Appendix 2.3.

3.6 Apply DESERT to More Protein Targets

To test the generalization ability of our method more widely, we also apply DESERT to the test data from Luo et al. [2], which contains 100 protein targets. As shown in Table 3, based on the idea of

structure determines properties, DESERT-POCKET generalizes well on different target proteins in both setting, i.e., with/without post-processing. Supervised methods, like 3D SBDD, hindered by scarce training data, can not generate diverse molecules. In contrast, training on massive unbound and drug-like molecules, DESERT easily generates diverse and promising molecules. Moreover, sketching molecular shapes based on given pockets also contributes to the better binding affinity of DESERT. However, DESERT gives a lower SA score than 3D SBDD. We assume that it is because the generated molecules of DESERT tend to be structurally complicated, which leads to a slightly worse synthesis score.

4 Conclusions

In this study, we propose a zero-shot drug design method DESERT, which splits the drug design process into two stages: sketching and generating. DESERT bridges the two stages with the molecular shape and utilize a large-scale molecular database to reduce the dependence on experimental data and docking simulation. Experiments show that DESERT achieves a new state-of-the-art at a fast speed.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. Both Hao Zhou and Xinyu Dai are the corresponding authors. This work is jointly supported by Guoqiang Research Institute General Project, Tsinghua University (No. 2021GQG1012) and National Science Foundation of China (No. 61936012 and 61976114).

References

[1] Xiliang Zheng et al. "Pocket-based drug design: exploring pocket space". In: *The AAPS journal* 15.1 (2013), pp. 228–241.

[2] Shitong Luo et al. "A 3D Generative Model for Structure-Based Drug Design". In: *Advances in Neural Information Processing Systems* 34 (2021).

[3] Pavol Drotár et al. "Structure-aware generation of drug-like molecules". In: *arXiv preprint arXiv:2111.04107* (2021).

[4] Leonardo G Ferreira et al. "Molecular docking and structure-based drug design strategies". In: *Molecules* 20.7 (2015), pp. 13384–13421.

[5] Xuan-Yu Meng et al. "Molecular docking: a powerful approach for structure-based drug discovery". In: *Current computer-aided drug design* 7.2 (2011), pp. 146–157.

[6] Nafisa M Hassan et al. "Protein-ligand blind docking using QuickVina-W with inter-process spatio-temporal integration". In: *Scientific reports* 7.1 (2017), pp. 1–13.

[7] Marco De Vivo et al. "Role of molecular dynamics and related methods in drug discovery". In: *Journal of medicinal chemistry* 59.9 (2016), pp. 4035–4061.

[8] Jacob D Durrant and J Andrew McCammon. "Molecular dynamics simulations and drug discovery". In: *BMC biology* 9.1 (2011), pp. 1–9.

[9] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. "Junction tree variational autoencoder for molecular graph generation". In: *International conference on machine learning*. PMLR. 2018, pp. 2323–2332.

[10] Yuwei Yang et al. *Knowledge Guided Geometric Editing for Unsupervised Drug Design*. 2022. URL: `https://openreview.net/forum?id=91muTwt1_t5`.

[11] Seokho Kang and Kyunghyun Cho. "Conditional molecular design with deep generative models". In: *Journal of chemical information and modeling* 59.1 (2018), pp. 43–52.

[12] Marwin HS Segler et al. "Generating focused molecule libraries for drug discovery with recurrent neural networks". In: *ACS central science* 4.1 (2018), pp. 120–131.

[13] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. "Grammar variational autoencoder". In: *International conference on machine learning*. PMLR. 2017, pp. 1945–1954.

[14] Qi Liu et al. "Constrained graph variational autoencoders for molecule design". In: *Advances in neural information processing systems* 31 (2018).

[15] Tengfei Ma, Jie Chen, and Cao Xiao. "Constrained generation of semantically valid graphs via regularizing variational autoencoders". In: *Advances in Neural Information Processing Systems* 31 (2018).

[16] Bidisha Samanta et al. "Nevae: A deep generative model for molecular graphs". In: *Journal of machine learning research. 2020 Apr; 21 (114): 1-33* (2020).

[17] Wengong Jin et al. "Learning multimodal graph-to-graph translation for molecular optimization". In: *arXiv preprint arXiv:1812.01070* (2018).

[18] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. "Composing molecules with multiple property constraints". In: *arXiv preprint arXiv:2002.03244* (2020).

[19] Ari Seff et al. "Discrete object generation with reversible inductive construction". In: *Advances in Neural Information Processing Systems* 32 (2019).

[20] Nicola De Cao and Thomas Kipf. "MolGAN: An implicit generative model for small molecular graphs". In: *arXiv preprint arXiv:1805.11973* (2018).

[21] Rafael G ó mez-Bombarelli et al. "Automatic chemical design using a data-driven continuous representation of molecules". In: *ACS central science* 4.2 (2018), pp. 268–276.

[22] Robin Winter et al. "Efficient multi-objective molecular optimization in a continuous latent space". In: *Chemical science* 10.34 (2019), pp. 8016–8024.

[23] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. "Multi-objective molecule generation using interpretable substructures". In: *International conference on machine learning*. PMLR. 2020, pp. 4849–4859.

[24] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. "Deep reinforcement learning for de novo drug design". In: *Science advances* 4.7 (2018), eaap7885.

[25] Jiaxuan You et al. "Graph convolutional policy network for goal-directed molecular graph generation". In: *Advances in neural information processing systems* 31 (2018).

[26] Chence Shi et al. "Graphaf: a flow-based autoregressive model for molecular graph generation". In: *arXiv preprint arXiv:2001.09382* (2020).

[27] Zhenpeng Zhou et al. "Optimization of molecules via deep reinforcement learning". In: *Scientific reports* 9.1 (2019), pp. 1–10.

[28] Jan H Jensen. "A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space". In: *Chemical science* 10.12 (2019), pp. 3567–3572.

[29] Sungsoo Ahn et al. "Guiding deep molecular optimization with genetic exploration". In: *Advances in neural information processing systems* 33 (2020), pp. 12008–12021.

[30] AkshatKumar Nigam et al. "Augmenting genetic algorithms with deep neural networks for exploring the chemical space". In: *arXiv preprint arXiv:1909.11655* (2019).

[31] R Vasundhara Devi, S Siva Sathya, and Mohane Selvaraj Coumar. "Evolutionary algorithms for de novo drug design–A survey". In: *Applied Soft Computing* 27 (2015), pp. 543–552.

[32] Jan H Jensen. "A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space". In: *Chemical science* 10.12 (2019), pp. 3567–3572.

[33] Yutong Xie et al. "Mars: Markov molecular sampling for multi-objective drug discovery". In: *arXiv preprint arXiv:2103.10432* (2021).

[34] Tianfan Fu et al. "Mimosa: Multi-constraint molecule sampling for molecule optimization". In: *arXiv preprint arXiv:2010.02318* (2020).

[35] Tomohide Masuda, Matthew Ragoza, and David Ryan Koes. "Generating 3D Molecular Structures Conditional on a Receptor Binding Site with Deep Generative Models". In: *CoRR* abs/2010.14442 (2020). arXiv: `2010.14442`. URL: `https://arxiv.org/abs/2010.14442`.

[36] Oleg Trott and Arthur J Olson. "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading". In: *Journal of computational chemistry* 31.2 (2010), pp. 455–461.

[37] Jerome Eberhardt et al. "AutoDock Vina 1.2. 0: New docking methods, expanded force field, and python bindings". In: *Journal of Chemical Information and Modeling* 61.8 (2021), pp. 3891–3898.

[38] Francois Berenger et al. "Lean-Docking: Exploiting Ligands' Predicted Docking Scores to Accelerate Molecular Docking". In: *Journal of Chemical Information and Modeling* 61.5 (2021), pp. 2341–2352.

[39] Ondrej Vavra et al. "CaverDock: a molecular docking-based tool to analyse ligand transport through protein tunnels and channels". In: *Bioinformatics* 35.23 (2019), pp. 4986–4993.

[40] David Ramrez and Julio Caballero. "Is it reliable to use common molecular docking methods for comparing the binding affinities of enantiomer pairs for their protein target?" In: *International journal of molecular sciences* 17.4 (2016), p. 525.

[41] Kun-Yi Hsin, Samik Ghosh, and Hiroaki Kitano. "Combining machine learning systems and multiple docking simulation packages to improve docking prediction reliability for network pharmacology". In: *PloS one* 8.12 (2013), e83922.

[42] Alan R Katritzky et al. "Computational Chemistry Approaches for Understanding how Structure Determines Properties". In: *Zeitschrift f ü r Naturforschung B* 64.6 (2009), pp. 773–777.

[43] RM Kell and PB Stickney. *How Structure Determines Properties of Plastics*. Tech. rep. BATTELLE MEMORIAL INST COLUMBUS OHIO, 1964.

[44] Alan R Katritzky and Dan C Fara. "How chemical structure determines physical, chemical, and technological properties: An overview illustrating the potential of quantitative structure-property relationships for fuels science". In: *Energy & Fuels* 19.3 (2005), pp. 922–935.

[45] Milan Rambukwella et al. "Ligand Structure Determines Nanoparticles' Atomic Structure, Metal-Ligand Interface and Properties". In: *Frontiers in chemistry* (2018), p. 330.

[46] Minkai Xu et al. "Learning neural generative dynamics for molecular conformation generation". In: *arXiv preprint arXiv:2102.10240* (2021).

[47] Sereina Riniker and Gregory A Landrum. "Better informed distance geometry: using what we know to improve conformation generation". In: *Journal of chemical information and modeling* 55.12 (2015), pp. 2562–2574.

[48] Stephan T Stern, Marilyn N Martinez, and David M Stevens. "When is it important to measure unbound drug in evaluating nanomedicine pharmacokinetics?" In: *Drug Metabolism and Disposition* 44.12 (2016), pp. 1934–1939.

[49] Miha Skalic et al. "Shape-based generative modeling for de novo drug design". In: *Journal of chemical information and modeling* 59.3 (2019), pp. 1205–1214.

[50] Ashutosh Kumar and Kam YJ Zhang. "Advances in the development of shape similarity methods and their application in drug discovery". In: *Frontiers in chemistry* 6 (2018), p. 315.

[51] Camila Cardoso Santos et al. "Drug screening using shape-based virtual screening and in vitro experimental models of cutaneous Leishmaniasis". In: *Parasitology* 148.1 (2021), pp. 98–104.

[52] Mu Gao and Jeffrey Skolnick. "A comprehensive survey of small-molecule binding pockets in proteins". In: *PLoS computational biology* 9.10 (2013), e1003302.

[53] Rob LM Van Montfort and Paul Workman. "Structure-based drug design: aiming for a perfect fit". In: *Essays in biochemistry* 61.5 (2017), pp. 431–437.

[54] Yaxia Yuan, Jianfeng Pei, and Luhua Lai. "Binding site detection and druggability prediction of protein targets for structure-based drug design". In: *Current pharmaceutical design* 19.12 (2013), pp. 2326–2333.

[55] Vincent Le Guilloux, Peter Schmidtke, and Pierre Tuffery. "Fpocket: an open source platform for ligand pocket detection". In: *BMC bioinformatics* 10.1 (2009), pp. 1–11.

[56] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[57] A van Bondi. "van der Waals volumes and radii". In: *The Journal of physical chemistry* 68.3 (1964), pp. 441–451.

[58] Stephan Zheng et al. "Improving the robustness of deep neural networks via stability training". In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2016, pp. 4480–4488.

[59] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[60] Elena S Salmina, Norbert Haider, and Igor V Tetko. "Extended functional groups (EFG): an efficient set for chemical characterization and structure-activity relationship studies of chemical compounds". In: *Molecules* 21.1 (2015), p. 1.

[61] Alexis Conneau et al. "Unsupervised cross-lingual representation learning at scale". In: *arXiv preprint arXiv:1911.02116* (2019).

[62] Jrg Degen et al. "On the Art of Compiling and Using'Drug-Like'Chemical Fragment Spaces". In: *ChemMedChem: Chemistry Enabling Drug Discovery* 3.10 (2008), pp. 1503–1507.

[63] Mohit Iyyer et al. "Adversarial Example Generation with Syntactically Controlled Paraphrase Networks". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 1875–1885. DOI: 10.18653/v1/N18-1170. URL: https://aclanthology.org/N18-1170.

[64] *Indicator function*. URL: https://en.wikipedia.org/wiki/Indicator_function.

[65] *Translation (geometry)*. URL: https://en.wikipedia.org/wiki/Translation_(geometry).

[66] *Quaternion*. URL: https://en.wikipedia.org/wiki/Quaternion.

[67] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[68] Jiangtao Feng et al. "ParaGen : A Parallel Generation Toolkit". In: *arXiv preprint arXiv:2210.03405* (2022).

[69] Ari Holtzman et al. "The curious case of neural text degeneration". In: *arXiv preprint arXiv:1904.09751* (2019).

[70] Binghong Chen et al. "Molecule Optimization by Explainable Evolution". In: *International Conference on Learning Representations*. 2020.

[71] David Ryan Koes and Carlos J Camacho. "Shape-based virtual screening with volumetric aligned molecular shapes". In: *Journal of computational chemistry* 35.25 (2014), pp. 1824–1834.

[72] Thomas S Rush et al. "A shape-based 3-D scaffold hopping method and its application to a bacterial protein- protein interaction". In: *Journal of medicinal chemistry* 48.5 (2005), pp. 1489–1495.

[73] Sunghwan Kim, Evan E Bolton, and Stephen H Bryant. "PubChem3D: shape compatibility filtering using molecular shape quadrupoles". In: *Journal of cheminformatics* 3.1 (2011), pp. 1–14.

[74] Roland J-M Pellenq et al. "A realistic molecular model of cement hydrates". In: *Proceedings of the National Academy of Sciences* 106.38 (2009), pp. 16102–16107.

[75] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[76] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

1 Method

1.1 Sketching

In Pocket-based Sketching, we present an algorithm to obtain the molecule shape, which is of the appropriate size and complementary to protein pockets. The algorithm uses a seed shape to intersect with the pocket gradually.

To obtain the seed shape, we first randomly sample several molecules from ZINC, then use the overlapping of their shapes as the seed shape. By using the overlapping strategy, we make the sketched pseudo molecular shapes more native-molecule-like and not overly dependent on one specific molecule. Algorithm 1 is the pseudo-code.

We set the volume threshold to 300Å^3, the average volume of molecules. The step size is the same as the voxel resolution, i.e., 0.5Å. We initially place the seed shape in a random position as long as the seed shape and the pocket shape do not overlap.

1.2 Generating

Pilot Experiments of Tokenizing Methods We evaluate different molecule tokenizing methods based on three principles (preserving functional groups, appropriate vocabulary size, and no circles structures) and propose our approach in the context of pre-training. There are several popular tokenizing methods for drug design: Atom-based [1, 2] shatters a molecule into atoms. Cut Single Bond [3] cuts all single bonds in a molecule. Both RECAP [4] and BRICS [5] use chemical reaction rules for preserving functional groups. The results in Table 1 show: (1) Atom-based and Cut Single Bond can not preserve functional groups. Because they do not distinguish whether a chemical bond belongs to a functional group, they ruin the molecular structures that are vital for determining molecule properties. (2) Two chemical-reaction-aware methods behave differently. As RECAP has more conservative rules, it produces a vocabulary at least one magnitude larger than its counterparts.

Because BRICS shows a better balance among these principles, we adapt it to serve our pre-trained model. Furthermore, because over 60% of the out-of-vocabulary problem, an essential factor affecting the pre-trained model quality [6], is caused by the combinations between rings and other structures through single bonds, we add an extra rule to BRICS: cut all single bonds attached to a ring.

Details of Discretization To stabilize the training process, we discretize two continuous variables, i.e., translation vector and rotation quaternion. Specifically, we map them into two discrete spaces.

For the translation vector, its discrete space is grids in 3D space (see Figure 1a). Briefly speaking, we represent this continuous vector with the discrete index of the grid. We represent the i-th translation bin as the coordinate of its centre $t_i^{\text{bin}} \in \mathbb{R}^3$. The discretization of any continuous translation operator $t \in \mathbb{R}^3$ can be computed by $\arg\min_i \|t_i^{\text{bin}}, t\|_2$. The total number of grids is $21,952$.

Because the rotation axis and rotation angle can determine the rotation quaternion, the discrete space of the quaternion contains two parts. As shown in Figure 1b, we first enumerate rotation axes (x, y, z) in 3D space, then for each axis, we list the rotation angle every θ degrees. We represent the i-th rotation bin (stands for rotating θ_i degrees around an axis (x_i, y_i, z_i)) as a quaternion

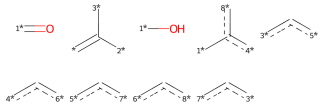
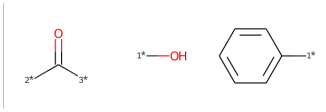
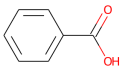
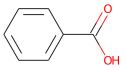
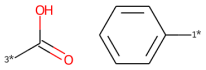
36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Algorithm 1 Pocket-based Sketching

Input: a function λ measuring the volume of a shape, a threshold t, a step size α
a pocket shape S_p located at C_p, a seed shape S_s located at C_s where $\lambda\left(\bigcap(S_p, S_s)\right) = 0$
Output: The sampled molecule shape S_d.

```
while True do
    V ← λ(⋂(S_p, S_s))
    if V ≥ t then
        S_d ← ⋂(S_p, S_s)
        break
    end if
    C_s = αC_p + (1 − α)C_s
end while
```

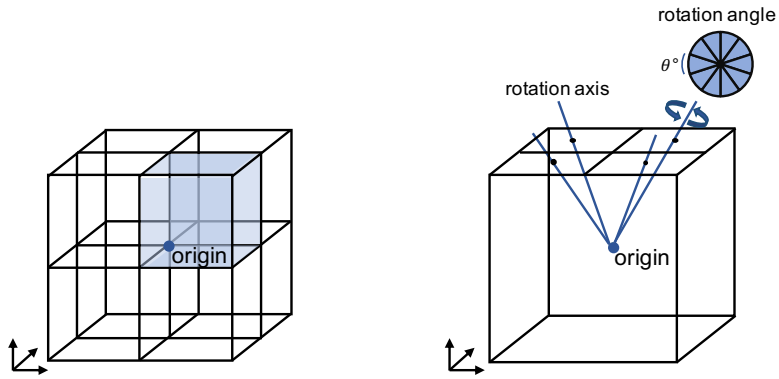
Table 1: Comparison of different tokenizing methods. Coverage is the percentage of novel molecules that the vocabulary can handle.

Method	Coverage	Vocabulary Size	Case
Atom-based	100.0	49	
Cut Single Bond	74.3	16,920	
RECAP	40.1	289,188	
BRICS	61.2	49,339	
Ours	70.4	23,896	

$q_i^{\text{bin}} = (x_i, y_i, z_i, \theta_i) \in \mathbb{R}^4$. The discretization of any continuous rotation operator $q \in \mathbb{R}^4$ can be computed by $\arg\min_i \|q_i^{\text{bin}}, q\|_2$. The total number of bins is $8,763$. To be precise, we enumerate 363 axes and 24 angles (i.e., 15 degrees per angle).

Another reason for the discretization is to avoid the discontinuity of quaternions when optimizing them [7, 8]. Specifically, there are several approaches to parameterize the rotation operator: quaternion [9], euler-angle [10], and SO(3) group (i.e., the rotation matrix) [11]. Quaternion and euler-angle are sometimes ambiguous and discontinuous. For example, the rotation operator is periodic, rotating $180°$ is equal to rotating $-180°$ and rotating $179.9°$ is very close to rotating $-179.9°$. Without discretization, we will excessively penalize the model according to the mean-square-error (i.e., $[179.9 - (-179.9)]^2 = 359.8^2$). While in this paper, we convert a regression problem into a classification one with discretization, thus avoiding the above issues. Instead of discretizing quaternions, AlphaFold [12] avoids such issues by regarding the quaternion as an intermediate variable and not optimizing the quaternion directly. We leave adopting AlphaFold's strategy for future work.

Greedy Algorithm for Connecting Fragments Algorithm 2 is responsible for converting the separated fragments into a complete 3D molecule. To be specific, we first place all the fragments in 3D space according to the predictions of SHAPE2MOL. Then, for each time, we greedily chose the



(a) Discrete space for translation vector (b) Discrete space for rotation quaternion

Figure 1: Discretization of translation vector and rotation quaternion. Both origins locate at the centroid of the fragment.

two closest breakpoints from different fragments and connect the fragments through the breakpoints. By repeating the process, the fragments get larger and larger. When there are not enough breakpoints to connect, we return the largest fragment as the final molecule. For potential residual breakpoints, we attach carbon atoms to them for molecular validity.

Algorithm 2 Greedy Algorithm for Connecting Fragments

Input: Fragment sequence $S = [(C_i, P_i, R_i)]_{i=0}^{l_s}$, fragment vocabulary V.
Output: A whole molecule M formed by the fragments.

```
B ← {}
F ← {}
for i ← 0 to l_s do
    (C_i, P_i, R_i) ← S[i]
    f ← query_vocabulary(V, C_i)
    f ← rotate(translate(f, P_i), R_i)          ▷ Place the fragment in 3D space
    b ← get_breakpoints(f)                      ▷ Return all the breakpoints in current fragment
    F ← F ∪ {f}
    B ← B ∪ {b}
end for
while |B| ≥ 2 do
    b_1, b_2 ← get_breakpoints(B)               ▷ Chose two nearest breakpoints from different fragments
    if b_1 is None or b_2 is None then
        break
    end if
    f_1 ← get_fragment(F, b_1)                  ▷ Retrieve the corresponding fragment of the breakpoint
    f_2 ← get_fragment(F, b_2)
    M_partial ← attach_fragments(f_1, b_1, f_2, b_2)   ▷ Connect fragments through the breakpoints
    B ← B \ {b_1, b_2}
    F ← F \ {f_1, f_2}
    F ← F ∪ {M_partial}
end while
M ← get_largest_fragment(F)                     ▷ Chose the largest fragment as the output molecule
if |B| ≠ 0 then
    M ← attach_carbon(M, B)                     ▷ Handle the remaining breakpoints by attaching carbon atoms
end if
```

1.3 More Details of SHAPE2MOL

Output of Shape Encoder The output of the shape encoder is the continuous representation of each 3D patch, which contain the geometric information of input molecular shape. It will serve as the context of the decoder to constrain the shape of generated molecules.

Input of Shape Decoder The input of the shape decoder at the time step t is the fragment category, rotation quaternion, and translation vector from the output of the previous step $t-1$. We use them to tell the model how exactly a fragment is placed in 3D space so that the model can generate the next fragment connected with it. The output of the shape encoder is fed into the decoder as the geometric context, through the cross-attention module.

Hyperparameters of SHAPE2MOL Both encoder and decoder of SHAPE2MOL have 12 stacked 8-head Transformer layers, whose model size is 1024 and feedforward dimension is 4096. We use ReLu as the activation function and set the position embedding learnable in the training process.

For the encoder, following liGAN, we set the resolution of the voxelized shape to 0.5Å and the side length of the spanned cube to 14Å. The size of 3D patches is $4 \times 4 \times 4$, which allows us to handle a large number of voxels. Specifically, we reshape $21,952$ voxels into 343 3D patches in the paper.

The decoder has 3 different embeddings, denoting the fragment category, the discretized rotation, and the discretized translation, respectively. Similarly, we also use 3 different output heads to predict the category, rotation, and translation. The output heads share their weights with the embeddings. The total number of model parameters is 650M.

2 Experiments

2.1 Metrics

- Uniqueness **(Uniq)** is the percentage of unique molecules among all generated results.
- Novelty **(Nov)** is the percentage of generated molecules with Tanimoto similarity [13] less than 0.4 compared to its nearest neighbor in existing ligands [14].
- Diversity **(Div)** is the internal diversity of the generated molecules and calculated as $\frac{2}{n(n-1)} \sum_{x \neq x'} 1 - sim(x, x')$.
- Success rate **(Succ)** is the percentage of generated molecules that pass the predefined thresholds for the desired properties. Specifically, we based on a widely used rule of thumb [15] to set the thresholds as $\mathrm{QED} \geq 0.25$, $\mathrm{SA_{score}} \geq 0.59$ and $\mathrm{Vina_{score}} \leq -8.18$ kcal/mol, where QED is the index of drug-likeness [16] and $\mathrm{SA_{score}}$ is the index of synthetic accessibility [17].
- Product **(Prod)** is the product of the four metrics above and serves as a comprehensive evaluation.
- Median Vina Score **(Median)** is the median of $\mathrm{Vina_{score}}$. $\mathrm{Vina_{score}}$ is the binding energy evaluation using the global energy optimization algorithm in Vina [18], which can reflect the binding affinity between proteins and molecules.

2.2 More Ablation Study of Generating

Model Variants As shown in Figure 3, we study two variants in the model design: discretization and robust training. The discretization consistently improves the result, while the continuous variant is worse. A possible reason is a non-linear relationship between quaternions and rotation angles. [1] We observe minor improvement for finer granularity and leave it for future exploration. We also study the ability of handling shape noise caused by a possibly large pocket size. The model trained with shape noise (i.e. robust training) shows better performance when test data are noisy, while the performance under standard training drops clearly.

[1] For example, if a molecule rotates 90° along an axis, the mean squared error of quaternion is 1; when the angle is 10°, the error is 0.98.

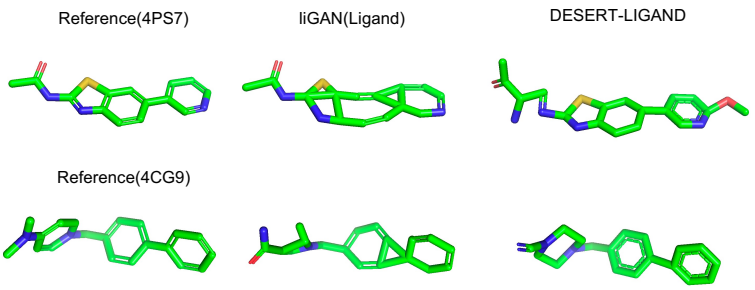


Figure 2: More cases from liGAN and DESERT.

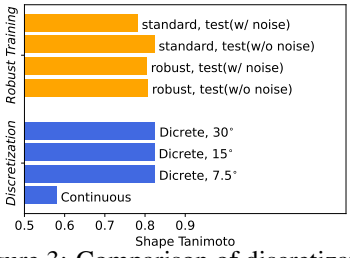


Figure 3: Comparison of discretization and robust training.

Table 2: Comparison of difference decoding strategy.

Sampling Method	Div	Prod	Median
Beam Search (Beam=10)	0.70	0.00	-5.87
Greedy Decoding	0.65	0.00	-5.83
Top K (K=10)	0.91	0.04	-5.99
Top P (P=0.95)	0.93	0.06	-6.01
+ post-processing	0.92	0.17	-7.34

Table 3: Combine protein chemical information.

Method	Prod	Median
DESERT-POCKET	0.55	-9.32
+ chemical(Weak)	0.53	-9.19
+ chemical(Strong)	0.52	-9.03

Table 4: Comparison of difference seed shape.

Seed Shape	Succ	Prod	Median
None	82.0	0.74	-8.85
Sphere	75.5	0.68	-9.13
Molecules	61.0	0.55	-9.32

Decoding Strategy We replace the Nucleus sampling method, i.e., Top P sampling, in the decoding strategy with Beam Search [19], Greedy Decoding [20], and Top K sampling [20] to study the influence of different sampling methods on the performance of DESERT. As shown in Table 2, Beam Search and Greedy Decoding are poor at providing diverse molecules, which suggests that the output probability is concentrated. They can only get several fragments with high output probability. When low probability fragments are obtained, the ranking mechanism in Beam Search drops them in the subsequent decoding step. Due to their internal annealing mechanism, both Top K and Top P can provide relatively diverse results.

Chemical Information Driven Design We also explore the potential of integrating chemical information of proteins into drug design. Briefly speaking, based on hydrogen bond acceptor-donor rules [21], we put the fragment with more hydrogen atoms into the pocket region with more oxygen, nitrogen, and fluorine atoms. The results in Table 3 show that the strategy does not achieve better performance when we increase the effect of chemical information on the output probability. The reason might be not considering detailed chemical information, like the bond length. We leave a more thoughtful method for combining the chemical information for future work.

Atom-based Pre-training We pre-train an atom-based SHAPE2MOL to analyze the contribution of our fragment-based generation style. Specifically, we keep most pre-training processes the same but replace our tokenizing method with the atom-based strategy (both mentioned in Section 1.2). As drug molecules mainly consist of carbon atoms, the atom-based SHAPE2MOL ignores input shapes and keeps outputting carbon atoms. The atom-based variant can not design any valid molecules, which indicates the fragment-based style is crucial for our pre-training process.

Invariance of SHAPE2MOL Following liGAN, when training SHAPE2MOL, we randomly rotate and translate the molecules to make our model have rotation and translation invariance ability. We compare the similarity of generated molecules based on different or the same molecular shapes (both are randomly rotated and translated). The similarity rises from 0.092 to 0.508, which shows that with the same molecular shape as input, the model produces similar molecules.

2.3 More Ablation Study of Sketching

Seed Shape Type We evaluate the influence of different seed shape in Table 4. We find that directly using the pocket as the shape may contribute to suboptimal result in providing highly active molecules. Through the sketching stage, we obtain more realistic and diverse molecular shapes. These shapes benefit the molecular activity but result in more complex structures and lower synthetic accessibility in generated molecules.

3 Discussion of Related Work

3.1 Shape-based Drug Design

Here we discuss the relationship between our method and existing shape-based drug design [22] approaches. Some previous work designed new drugs based on the shape of a known ligand. Traditional approaches work in a retrieval way – finding molecules whose shape is most similar to a known one [23, 24]. Modern deep learning models can decode a molecule from its shape [1, 25]. Such ligand-based generation can not generalize to unseen pockets. Luo et al. [2] directly generates molecules from the pocket shape, which is mostly close to our work. However, our model works in a fragment-based fashion, while theirs works in an atomic way. What makes our model especially different is that we utilize the power of the pre-training model to make pocket-based drug design more promising.

3.2 Tokenization and Linearization

Here we discuss how the tokenization and linearization procedures relate to fragment-based drug design (FBDD) [26, 27].

Fragment-Based Drug Design Briefly, there are two approaches in FBDD: growing the fragment synthetically to a proximal binding site or by linking two fragments together [27]. Our method can be classified as the previous type since the linearization generates a molecule in a one-by-one fashion.

Tokenization The procedure is carefully designed for deep generative models to avoid loops and preserve functionalities, which is in spirit to the principle of [3]. Some FBDD work, such as [28] works in a discriminative approach. Thus there are not many constraints when cutting the molecules. Podda et al. [29] uses the SMILES-fragment rather than a real molecule-fragment. Thus it can not utilize rich structured features.

Linearization The procedure aims at traversing (or generating) a structured object in a left-to-right approach [30], which is tractable and scalable. It is borrowed from the area of computational linguistics, more specifically, syntactic parsing and structured generation [31, 32]. For micro-molecules, linearization (such as SMILES [33]) has been adopted for several decades. There is a line of research work for SMILES-based generation [34, 35, 36]. Similarly, in the area of macro-molecule, Huang et al. [37] designs a linear structure for estimating the likelihood of the structure of RNA.

Comparison Compared with traditional linearized sequences such as SMILES [33], our method utilizes structural information to segment the molecules to preserve their functionality. Compared with topological generation based on a graph [3, 38, 39], our method is more scalable to big data since generating the variable graph topology is not friendly to large-batch training in neural networks.

4 More Cases

We show more cases from liGAN and DESERT in Figure 2. Similarly, liGAN struggles to produce realistic molecular structures, while DESERT generates better results.

References

[1] Tomohide Masuda, Matthew Ragoza, and David Ryan Koes. "Generating 3D Molecular Structures Conditional on a Receptor Binding Site with Deep Generative Models". In: *CoRR* abs/2010.14442 (2020). arXiv: 2010.14442. URL: https://arxiv.org/abs/2010.14442.

[2] Shitong Luo et al. "A 3D Generative Model for Structure-Based Drug Design". In: *Advances in Neural Information Processing Systems* 34 (2021).

[3] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. "Junction tree variational autoencoder for molecular graph generation". In: *International conference on machine learning*. PMLR. 2018, pp. 2323–2332.

[4] Xiao Qing Lewell et al. "Recap retrosynthetic combinatorial analysis procedure: a powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry". In: *Journal of chemical information and computer sciences* 38.3 (1998), pp. 511–522.

[5] Jrg Degen et al. "On the Art of Compiling and Using'Drug-Like'Chemical Fragment Spaces". In: *ChemMedChem: Chemistry Enabling Drug Discovery* 3.10 (2008), pp. 1503–1507.

[6] Wen Tai et al. "exBERT: Extending pre-trained models with domain-specific vocabulary under constrained training resources". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 1433–1439.

[7] Yi Zhou et al. "On the continuity of rotation representations in neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5745–5753.

[8] Luca Falorsi et al. "Explorations in homeomorphic variational auto-encoding". In: *arXiv preprint arXiv:1807.04689* (2018).

[9] Wikipedia contributors. *Quaternion — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Quaternion&oldid=1112663286. [Online; accessed 4-October-2022]. 2022.

[10] Wikipedia contributors. *Euler angles — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Euler_angles&oldid=1112943781. [Online; accessed 4-October-2022]. 2022.

[11] Wikipedia contributors. *3D rotation group — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=3D_rotation_group&oldid=1101034704. [Online; accessed 4-October-2022]. 2022.

[12] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[13] David Rogers and Mathew Hahn. "Extended-connectivity fingerprints". In: *Journal of chemical information and modeling* 50.5 (2010), pp. 742–754.

[14] Marcus Olivecrona et al. "Molecular de-novo design through deep reinforcement learning". In: *Journal of cheminformatics* 9.1 (2017), pp. 1–14.

[15] Oleg Ursu et al. "DrugCentral 2018: an update". In: *Nucleic acids research* 47.D1 (2019), pp. D963–D970.

[16] G Richard Bickerton et al. "Quantifying the chemical beauty of drugs". In: *Nature chemistry* 4.2 (2012), pp. 90–98.

[17] Peter Ertl and Ansgar Schuffenhauer. "Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions". In: *Journal of cheminformatics* 1.1 (2009), pp. 1–11.

[18] Oleg Trott and Arthur J Olson. "AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading". In: *Journal of computational chemistry* 31.2 (2010), pp. 455–461.

[19] Markus Freitag and Yaser Al-Onaizan. "Beam search strategies for neural machine translation". In: *arXiv preprint arXiv:1702.01806* (2017).

[20] Ari Holtzman et al. "The curious case of neural text degeneration". In: *arXiv preprint arXiv:1904.09751* (2019).

[21] Roderick E Hubbard and Muhammad Kamran Haider. "Hydrogen bonds in proteins: role and strength". In: *eLS* (2010).

[22] Rob LM Van Montfort and Paul Workman. "Structure-based drug design: aiming for a perfect fit". In: *Essays in biochemistry* 61.5 (2017), pp. 431–437.

[23] Ashutosh Kumar and Kam YJ Zhang. "Advances in the development of shape similarity methods and their application in drug discovery". In: *Frontiers in chemistry* 6 (2018), p. 315.

[24] Camila Cardoso Santos et al. "Drug screening using shape-based virtual screening and in vitro experimental models of cutaneous Leishmaniasis". In: *Parasitology* 148.1 (2021), pp. 98–104.

[25] Miha Skalic et al. "Shape-based generative modeling for de novo drug design". In: *Journal of chemical information and modeling* 59.3 (2019), pp. 1205–1214.

[26] Philine Kirsch et al. "Concepts and core principles of fragment-based drug design". In: *Molecules* 24.23 (2019), p. 4309.

[27] Christopher W Murray and David C Rees. "The rise of fragment-based drug discovery". In: *Nature chemistry* 1.3 (2009), pp. 187–192.

[28] Harrison Green, David R Koes, and Jacob D Durrant. "DeepFrag: a deep convolutional neural network for fragment-based lead optimization". In: *Chemical Science* 12.23 (2021), pp. 8036–8047.

[29] Marco Podda, Davide Bacciu, and Alessio Micheli. "A deep generative model for fragment-based molecule generation". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2240–2250.

[30] Yijia Liu et al. "Transition-based syntactic linearization". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 113–122.

[31] Yoon Kim et al. "Unsupervised recurrent neural network grammars". In: *arXiv preprint arXiv:1904.03746* (2019).

[32] Oriol Vinyals et al. "Grammar as a foreign language". In: *Advances in neural information processing systems* 28 (2015).

[33] David Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *Journal of chemical information and computer sciences* 28.1 (1988), pp. 31–36.

[34] Matt J Kusner, Brooks Paige, and Jos é Miguel Hern á ndez-Lobato. "Grammar variational autoencoder". In: *International conference on machine learning*. PMLR. 2017, pp. 1945–1954.

[35] Hanjun Dai et al. "Syntax-directed variational autoencoder for structured data". In: *arXiv preprint arXiv:1802.08786* (2018).

[36] Seokho Kang and Kyunghyun Cho. "Conditional molecular design with deep generative models". In: *Journal of chemical information and modeling* 59.1 (2018), pp. 43–52.

[37] Liang Huang et al. "LinearFold: linear-time approximate RNA folding by 5'-to-3'dynamic programming and beam search". In: *Bioinformatics* 35.14 (2019), pp. i295–i304.

[38] Binghong Chen et al. "Molecule Optimization by Explainable Evolution". In: *International Conference on Learning Representations*. 2020.

[39] Yutong Xie et al. "Mars: Markov molecular sampling for multi-objective drug discovery". In: *arXiv preprint arXiv:2103.10432* (2021).