# A 3D Generative Model for Structure-Based Drug Design

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

We study a fundamental problem in structure-based drug design — generating molecules that bind to specific protein binding sites. While we have witnessed the great success of deep generative models in drug design, the existing methods are mostly string-based or graph-based, which attempt to model molecules from an existing distribution of SMILES strings or molecule graphs. Therefore, they are limited by the lack of spatial information and thus unable to be applied to structure-based design tasks. Particularly, such models have no or little knowledge of how molecules interact with their target proteins exactly in 3D space. In this paper, we propose a 3D generative model that generates molecules given a designated 3D protein binding pocket or site. Specifically, given a binding site as the 3D context, our model estimates the probability density of atom's occurrences in 3D space — positions that are more likely to have atoms will be assigned higher probability. To generate molecules, we propose an auto-regressive sampling scheme — atoms are sampled sequentially from the learned distribution until there is no room for new atoms inside the binding pocket. Combined with this sampling scheme, our model can generate valid and diverse molecules, which would be applicable to various structure-based molecular design tasks, such as molecule sampling and linker design. Experimental results demonstrate that molecules sampled from our model exhibit high binding affinity to specific targets and good drug properties such as drug-likeness even if the model is not explicitly optimized for them.

## 1 Introduction

Designing molecules that bind to a specific protein binding site, also known as structure-based drug design, is one of the most challenging tasks in drug discovery [2]. Searching for suitable molecule candidates *in silico* usually involves massive computational efforts because of the enormous space of synthetically feasible chemicals [24] and conformational degree of freedom of both compound and protein structures [13].

In recent years, we have witnessed the success of machine learning approaches to problems in drug design, especially on molecule generation. Most of these approaches use deep generative models to propose drug candidates by learning the underlying distribution of desirable molecules. However, most of such methods are generally SMILES/string-based [12, 18] or graph-based [19, 20, 15, 16]. They are limited by the lack of spatial information and unable to perceive how molecules interact with proteins in 3D space. Hence, these methods are not applicable to generating molecules that fit to a specific protein structure which is also known as the drug target. Another line of work studies generating molecules directly in 3D space [8, 30, 31, 21]. Most of them [8, 30, 31] can only handle very small organic molecules, not sufficient to generate drug-scale molecules which usually contain dozens of heavy atoms. [21] proposes to generate voxelized molecular images and use a

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

post-processing algorithm to reconstruct molecular structures. Though this method could produce drug-scale molecules for specific protein pockets, the quality of the sampling is heavily limited by voxelization. Therefore, generating high-quality drug molecules for specific 3D protein binding sites remains challenging.

In this work, we propose a 3D generative model to approach this task. Specifically, we aim at modeling the distribution of atom occurrence in the 3D space of the binding site. Formally, given a binding site $\mathcal{C}$ as input, we model the distribution $p(e, \boldsymbol{r}|\mathcal{C})$, where $\boldsymbol{r} \in \mathbb{R}^3$ is an arbitrary 3D coordinate and $e$ is atom type. To realize this distribution, we design a neural network architecture which takes as input a query 3D coordinate $\boldsymbol{r}$, conditional on the 3D context $\mathcal{C}$, and outputs the probability of $\boldsymbol{r}$ being occupied by an atom of a particular chemical element. In order to ensure the distribution is equivariant to $\mathcal{C}$'s rotation and translation, we utilize graph neural networks to perceive the context of each query coordinate.

Despite having a powerful neural network to model the distribution of atom occurrence $p(e, \boldsymbol{r}|\mathcal{C})$, how to generate *valid* and *diverse* molecules still remains technically challenging, mainly for the following two reasons: First, simply drawing *i.i.d.* samples from the distribution $p(e, \boldsymbol{r}|\mathcal{C})$ does not yield valid molecules because atoms within a molecule are not independent of each other. Second, a desirable sampling algorithm should capture the multi-modality of the feasible chemical space, i.e. it should be able to generate a diverse set of desired molecules given a specific binding context. To tackle the challenge, we propose an auto-regressive sampling algorithm. In specific, we start with a context consisting of only protein atoms. Then, we iteratively sample one atom from the distribution at each step and add it to the context to be used in the next step, until there is no room for new atoms. Compared to other recent methods [21, 25], our auto-regressive algorithm is simpler and more advantageous. It does not rely on post-processing algorithms to infer atom placements from density. More importantly, it is capable of multi-modal sampling by the nature of auto-regressive, avoiding additional latent variables via VAEs [17] or GANs [10] which would bring about extra architectural complexity and training difficulty.

We conduct extensive experiments to evaluate our approach. Quantitative and qualitative results show that: (1) our method is able to generate diverse drug-like molecules that have high binding affinity to specific targets based on 3D structures of protein binding sites; (2) our method is able to generate molecules with fairly high drug-likeness score (QED) [4] and synthetic accessibility score (SA) [6] even if the model is not specifically optimized for them; (3) in addition to molecule generation, the proposed method is also applicable to other relevant tasks such as linker design.

## 2 Related Work

**SMILES-Based and Graph-Based Molecule Generation** Deep generative models have been prevalent in molecule design. The overall idea is to use deep generative models to propose molecule candidates by learning the underlying distribution of desirable molecules. The majority of existing works can be roughly divided into two classes — string-based and graph-based. String-based methods represent molecules as linear strings, e.g. SMILES strings [34], making a wide range of language modeling tools readily applicable. For example, [5, 12, 28] utilize recurrent neural networks to learn a language model of SMILES string. However, string-based methods are prone to fail as strings are not always convertible to molecular graphs. In contrast, graph representation is more natural, and graph-based approaches thrive in recent years. The majority of graph-based models generate molecules in an auto-regressive fashion, i.e., adding atoms or fragments sequentially. They are generally built upon VAEs [15], normalizing flows [29], and reinforcement learning [35]. More recently, graph-based models focus on generating molecules that satisfy multiple property constraints [22, 16]. Despite the progress made in string-based and graph-based approaches, they are limited by the lack of spatial information and thus unable to be applied to structure-based drug design tasks [2]. Specifically, as 1D/2D-based methods, they are unable to perceive how molecules interact with their target proteins exactly in 3D space.

**Molecule Generation in 3D Space** There has been another line of methods that generate molecules directly in 3D space. [8] proposes an auto-regressive model. It takes a partially generated molecule as input and outputs the next atom's chemical element and the distances to previous atoms. Then, the atom is placed in the 3D space according to the constraint of distances. [30, 31] approach this task by reinforcement learning. They also generate 3D molecules sequentially. Different from [8], they

do not need training data. Instead, they rely on a reward function derived from the potential energy function of atomic systems. These works successfully generate *realistic* 3D molecules. However, they can only handle small organic molecules, not sufficient to generate drug-scale molecules which usually contain dozens of heavy atoms.

[21, 25] propose a non-autoregressive approach to 3D molecular generation which is able to generate *drug-scale* molecules. It represents molecules as 3D images by voxelizing molecules onto 3D meshgrids. In this way, the molecular generation problem is transformed into an image generation problem, making it possible to leverage sophisticated image generation techniques. In specific, it employs convolutional neural network-based VAEs to generate such molecular images. It also attempts to fuse the binding site structures into the generative network, enabling the model to generate molecules for designated binding targets. In order to reconstruct the molecular structure from the image, it leverages a post-processing algorithm to search for atom placements that best fit the image. Though this method successfully generates drug-scale 3D molecules, its generation quality is heavily limited by the following drawbacks: First, it is hardly scalable to large binding pockets, as the number of voxels grows cubically to the pocket's size. Second, the 3D molecular images' resolution is limited also due to the scalability issue, leading to limited precision. Third, conventional CNNs are not rotation-equivariant, which is crucial for modeling molecular systems. As a result, the model is prone to produce unrealistic molecules.

# 3 Method

Our goal is to generate a set of atoms that is able to form a valid drug-like molecule fitting to a specific binding site. To this end, we first present a 3D generative model in Section 3.1 that predicts the probability of atom occurrence in 3D space of the binding site. Second, we present in Section 3.2 the auto-regressive sampling algorithm for generating valid and multi-modal molecules from the model. Finally, in Section 3.3, we derive the training objective, by which the model learns to predict where should be placed and atoms and what type of atom should be placed.

## 3.1 3D Generative Model Design

A binding site can be defined as a set of atoms $\mathcal{C} = \{(\boldsymbol{a}_i, \boldsymbol{r}_i)\}_{i=1}^{N_b}$, where $N_b$ is the number of atoms in the binding site, $\boldsymbol{a}_i$ is the $i$-th atom's attributes such as chemical element, belonging amino acid, etc., and $\boldsymbol{r}_i$ is its 3D coordinate. To generate atoms in the binding site, we consider modeling the probability of atom occurring at some position $\boldsymbol{r}$ in the site. Formally, this is to model the density $p(e|\boldsymbol{r}, \mathcal{C})$, where $\boldsymbol{r} \in \mathbb{R}^3$ is an arbitrary 3D coordinate, and $e \in \mathcal{E} = \{\mathrm{H}, \mathrm{C}, \mathrm{O}, \ldots\}$ is the chemical element. Intuitively, this density can be interpreted as a classifier that takes as input a 3D coordinate $\boldsymbol{r}$ conditional on $\mathcal{C}$ and predicts the probability of $\boldsymbol{r}$ being occupied by an atom of type $e$.

To model $p(e|\boldsymbol{r}, \mathcal{C})$, we devise a model consisting of two parts: **Context Encoder** learns the representation of each atom in the context $\mathcal{C}$ via graph neural networks. **Spatial Classifier** takes as input a query position $\boldsymbol{r}$, then aggregates the representation of contextual atoms nearby it, and finally predicts $p(e|\boldsymbol{r}, \mathcal{C})$. The implementation of these two parts is detailed as follows.

**Context Encoder** The purpose of the context encoder is to extract information-rich representations for each atom in $\mathcal{C}$. We assume a desirable representation should satisfy two properties: (1) **context-awareness**: the representation of an atom should not only encode the property of the atom itself, but also encode its context. (2) **roto-translation invariance**: since the physical and biological properties of the system do not change according to rigid transforms, the representations that reflect these properties should be invariant to rigid transforms as well. To this end, we employ graph neural networks as the backbone of the context encoder, described as follows.

First of all, since there is generally no natural topology in $\mathcal{C}$, we construct a $k$-nearest-neighbor graph based on inter-atomic distances, denoted as $\mathcal{G} = \langle \mathcal{C}, \boldsymbol{A} \rangle$, where $\boldsymbol{A}$ is the adjacency matrix. We also denote the $k$-NN neighborhood of atom $i$ as $N_k(\boldsymbol{r}_i)$ for convenience. The context encoder will take $\mathcal{G}$ as input and output structure-aware node embeddings.

The first layer of the encoder is a simple linear layer. It maps atomic attributes $\{\boldsymbol{a}_i\}$ to initial embeddings $\{\boldsymbol{h}_i^{(0)}\}$. Then, these embeddings along with the graph structure $\boldsymbol{A}$ are fed into $L$ message

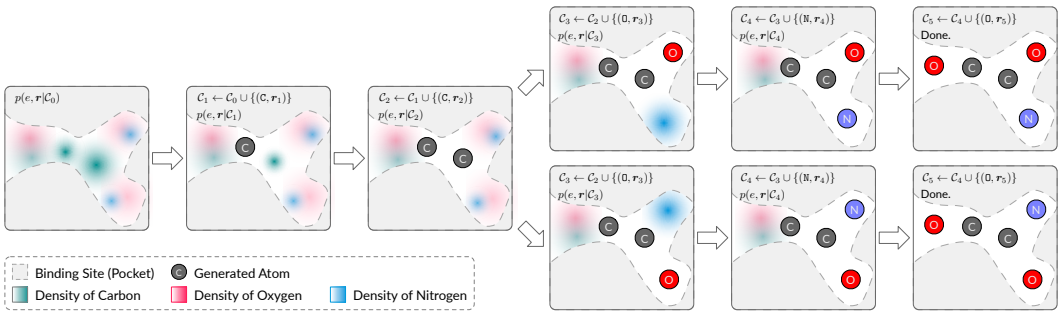


Figure 1: An illustration of the sampling process. Atoms are sampled sequentially. The probability density changes as we place new atoms. The sampling process naturally diverges, leading to different samples.

passing layers. Specifically, the formula of message passing takes the form:

$$\boldsymbol{h}_i^{(\ell+1)} = \boldsymbol{W}_0^{\ell} \boldsymbol{h}_i^{(\ell)} + \sum_{j \in N_k(\boldsymbol{r}_i)} \boldsymbol{W}^{\ell}(d_{ij}) \boldsymbol{h}_j^{(\ell)}, \tag{1}$$

where $\boldsymbol{W}$ is learnable weight matrix and $d_{ij}$ denotes the distance between atom $i$ and atom $j$. Here, the weight of message from $j$ to $i$ depends only on $d_{ij}$, ensuring its invariance to rotation and translation [27, 9]. Finally, we obtain $\{\boldsymbol{h}_i^{(L)}\}$ a set of embeddings for each atom in $\mathcal{C}$.

**Spatial Classifier** The spatial classifier takes as input a query position $\boldsymbol{r} \in \mathbb{R}^3$ and predicts the type of atom occupying $\boldsymbol{r}$. In order to make successful predictions, the model should be able to perceive the context around $\boldsymbol{r}$. Therefore, the first step of this part is to aggregate atom embeddings from the context encoder:

$$\boldsymbol{v} = \sum_{j \in N_k(\boldsymbol{r})} \boldsymbol{W}_{\text{aggr}}(\|\boldsymbol{r} - \boldsymbol{r}_j\|) \boldsymbol{h}_j^{(L)}, \tag{2}$$

where $N_k(\boldsymbol{r})$ is the $k$-nearest neighborhood of $\boldsymbol{r}$. Note that we weight different embedding using $\boldsymbol{W}_{\text{aggr}}$ according to distances because it is necessary to distinguish the contribution of different atoms in the context. Finally, in order to predict $p(e|\boldsymbol{r}, \mathcal{C})$, the aggregated feature $\boldsymbol{v}$ is then passed to a classical multi-layer perceptron classifier:

$$\boldsymbol{c} = \text{MLP}(\boldsymbol{v}), \tag{3}$$

where $\boldsymbol{c}$ is the non-normalized probability of chemical elements. The estimated probability of position $\boldsymbol{r}$ being occupied by atom of type $e$ is:

$$p(e|\boldsymbol{r}, \mathcal{C}) = \frac{\exp(\boldsymbol{c}[e])}{1 + \sum_{e' \in \mathcal{E}} \exp(\boldsymbol{c}[e'])}, \tag{4}$$

where $\mathcal{E}$ is the set of possible chemical elements. Unlike typical classifiers that apply softmax to $\boldsymbol{c}$, we make use of the extra degree of freedom by adding 1 to the denominator, so that the probability of "nothing" can be expressed as:

$$p(\texttt{Nothing}|\boldsymbol{r}, \mathcal{C}) = \frac{1}{1 + \sum \exp(\boldsymbol{c}[e'])}. \tag{5}$$

### 3.2 Sampling

Sampling a molecule amounts to generating a set of atoms $\{(e_i, \boldsymbol{r}_i)\}_{i=1}^{N_a}$. However, formulating an effective sampling algorithm is non-trivial because of the following three challenges. First, we have to define the joint distribution of $e$ and $\boldsymbol{r}$, i.e. $p(e, \boldsymbol{r}|\mathcal{C})$, from which we can jointly sample an atom's chemical element and its position. Second, notice that simply drawing i.i.d. samples from $p(e, \boldsymbol{r}|\mathcal{C})$ doesn't make sense because atoms are clearly not independent of each other. Thus, the sampling algorithm should be able to attend to the dependencies between atoms. Third, the sampling algorithm should produce multi-modal samples. This is important because in reality there is usually more than one molecule that can bind to a specific target.

In the following, we first define the joint distribution $p(e, \boldsymbol{r}|\mathcal{C})$. Then, we present an auto-regressive sampling algorithm to tackle the second and the third challenges.

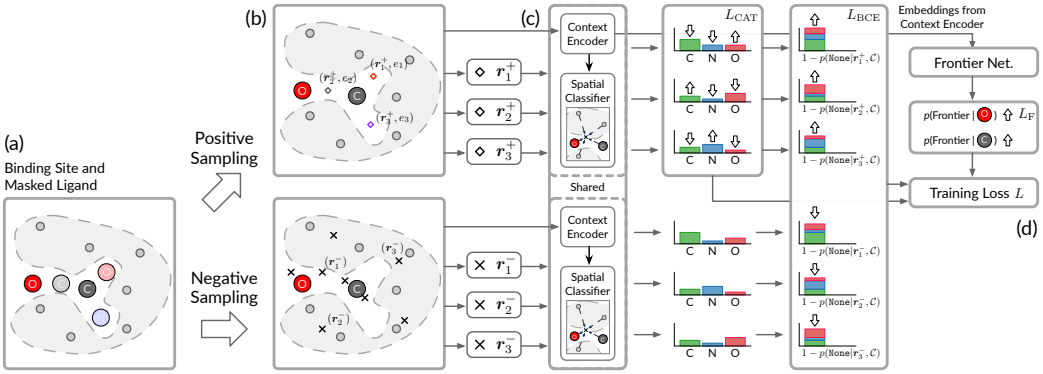


Figure 2: (a) A portion of the molecule is masked. (b) Positive coordinates are drawn from the masked atoms' positions and negative coordinates are drawn from the ambient space. (c) Both positive and negative coordinates are fed into the model. The model predicts the probability of atom occurrence at the coordinates. (d) Training losses are computed based on the discrepancy between predicted probabilities and ground truth.

**Joint Distribution** We define the joint distribution of coordinate $\boldsymbol{r}$ and atom type $e$ using Eq.4:

$$p(e, \boldsymbol{r}|\mathcal{C}) = \frac{\exp\left(\boldsymbol{c}[e]\right)}{Z}, \tag{6}$$

where $Z$ is an unknown normalizing constant and $\boldsymbol{c}$ is a function of $\boldsymbol{r}$ and $\mathcal{C}$ as described in the previous section. This is also known as a joint energy model (JEM) that interprets classifiers as energy-based models [11]. Though $p(e, \boldsymbol{r})$ is a non-normalized distribution, drawing samples from it would be efficient because the dimension of $\boldsymbol{r}$ is only 3. Viable sampling methods include Markov chain Monte Carlo (MCMC) or discretization.

**Auto-Regressive Sampling** We sample a molecule by progressively sampling one atom at each step. In specific, at step $t$, the context $\mathcal{C}_t$ contains not only protein atoms but also $t$ atoms sampled beforehand. Sampled atoms in $\mathcal{C}_t$ are treated equally as protein atoms in the model, but they have different attributes in order to differentiate themselves from protein atoms. Then, the $(t+1)$-th atom will be sampled from $p(e, \boldsymbol{r}|\mathcal{C}_t)$ and will be added to $\mathcal{C}_t$, leading to the context for next step $\mathcal{C}_{t+1}$. The sampling process is illustrated in Figure 1. Formally, we have:

$$\begin{aligned} &(e_{t+1}, \boldsymbol{r}_{t+1}) \sim p(e, \boldsymbol{r}|\mathcal{C}_t), \\ &\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup \{(e_{t+1}, \boldsymbol{r}_{t+1})\}. \end{aligned} \tag{7}$$

To determine when the auto-regressive sampling should stop, we employ an auxiliary network. The network takes as input the embedding of previously sampled atoms, and classifies them into two categories: frontier and non-frontier. If all the existing atoms are non-frontier, which means there is no room for more atoms, the sampling will be terminated. Finally, we use OpenBabel [23] to obtain bonds of generated structures.

In summary, the proposed auto-regressive algorithm succeeds to settle the aforementioned two challenges. First, the model is aware of other atoms when placing new atoms, thus being able to consider the dependencies between them. Second, auto-regressive sampling is a stochastic process. Its sampling path naturally diverges, leading to diverse samples.

### 3.3 Training

As we adopt auto-regressive sampling strategies, we propose a mask-predict training scheme — at training time, a random portion of the target molecule is masked, and the network learns to predict the masked part from the observable part and the binding site. This emulates the sampling process where the model can only observe partial molecules. The training loss consists of three terms described below.

First, to make sure the model is able to predict positions that actually have atoms (positive positions), we include a binary cross entropy loss to contrast positive positions against negative positions:

$$L_{\text{BCE}} = -\mathbb{E}_{\boldsymbol{r}\sim p_+}\left[\log\left(1 - p(\texttt{Nothing}|\boldsymbol{r}, \mathcal{C})\right)\right] - \mathbb{E}_{\boldsymbol{r}\sim p_-}\left[\log p(\texttt{Nothing}|\boldsymbol{r}, \mathcal{C})\right]. \tag{8}$$

Here, $p_+$ is a positive sampler that yields coordinates of masked atoms. $p_-$ is a negative sampler that yields random coordinates in the ambient space. $p_-$ is empirically defined as a Gaussian mixture model containing $|\mathcal{C}|$ components centered at each atom in $\mathcal{C}$. The standard deviation of each component is set to 2Å in order cover to the ambient space. Intuitively, the first term in Eq.8 increases the likelihood of atom placement for positions that should get an atom. The second term decreases the likelihood for other positions.

Second, our model should be able to predict the chemical element of atoms. Hence, we further include a standard categorical cross entropy loss:

$$L_{\text{CAT}} = -\mathbb{E}_{(e,\boldsymbol{r})\sim p_+}\left[\log p(e|\boldsymbol{r}, \mathcal{C})\right]. \tag{9}$$

Third, as introduced in Section 3.2, the sampling algorithm requires a frontier network to tell whether the sampling should be terminated. This leads to the last term — a standard binary cross entropy loss for training the frontier network:

$$L_{\text{F}} = \sum_{i\in\mathcal{F}\subseteq\mathcal{C}} \log\sigma(F(\boldsymbol{h}_i)) + \sum_{i\notin\mathcal{F}\subseteq\mathcal{C}} \log(1 - \sigma(F(\boldsymbol{h}_i))), \tag{10}$$

where $\mathcal{F}$ is the set of frontier atoms in $\mathcal{C}$, $\sigma$ is the sigmoid function, and $F(\cdot)$ is the frontier network that takes atom embedding as input and predicts the logit probability of the atom being a frontier. During training, an atom is regarded as a frontier if and only if (1) the atom is a part of the target molecule, and (2) at least one of its bonded atom is masked.

Finally, by summing up $L_{\text{BCE}}$, $L_{\text{CAT}}$, and $L_{\text{F}}$, we obtain the full training loss $L = L_{\text{BCE}} + L_{\text{CAT}} + L_{\text{F}}$. The full training process is illustrated in Figure 2.

# 4 Experiments

We evaluate the proposed method on two relevant structure-based drug design tasks: (1) **Molecule Design** is to generate molecules for given binding sites (Section 4.1), and (2) **Linker Prediction** is to generate substructures to link two given fragments in the binding site. (Section 4.2). Below, we describe common setups shared across tasks. Detailed task-specific setups are provided in each subsection.

**Data** We use the CrossDocked dataset [7] following Masuda et al. [21]. The dataset originally contains 22.5 million docked protein-ligand pairs at different levels of quality. We filter out data points whose binding pose RMSD is greater than 1Å, leading to a refined subset consisting of 184,057 data points. We randomly draw 150,000 data points from the subset for training, and randomly select 100 proteins that do *not appear* in the training split for testing.

**Model** We trained a universal model for all the tasks. The number of message passing layers in context encoder $L$ is 6, and the hidden dimension is 256. We train the model using the Adam optimizer at learning rate 0.0001. The training takes 48 hours on a single Nvidia 2080ti GPU. Other details about model architectures and training parameters are provided in the supplementary material. *The codes and datasets will be made publicly available once this paper gets published.*

## 4.1 Molecule Design

In this task, we generate molecules for specific binding sites with our model and baselines. The input to models are binding sites extracted from the proteins in the testing set. We sample 100 unique molecules for each target.

**Baselines** We compare our approach with the state-of-the-art baseline liGAN [21]. liGAN is based on conventional 3D convolutional neural networks. It generates voxelized molecular images and relies on a post-processing algorithm to reconstruct the molecule from the generated image.

Table 1: Mean and median values of the metrics on generation quality. (↑) indicates higher is better. (↓) indicates lower is better.

| Metric | | liGAN | **Ours** | Ref |
|---|---|---|---|---|
| Vina Score (kcal/mol, ↓) | Avg. | -6.144 | **-6.344** | -7.158 |
| | Med. | -6.100 | **-6.200** | -6.950 |
| QED (↑) | Avg. | 0.371 | **0.525** | 0.484 |
| | Med. | 0.369 | **0.519** | 0.469 |
| SA (↑) | Avg. | 0.591 | **0.657** | 0.733 |
| | Med. | 0.570 | **0.650** | 0.745 |
| High Affinity (%, ↑) | Avg. | 23.77 | **29.09** | - |
| | Med. | 11.00 | **18.50** | - |
| Diversity (↑) | Avg. | 0.655 | **0.720** | - |
| | Med. | 0.676 | **0.736** | - |

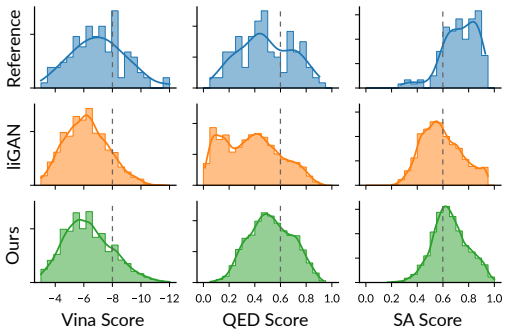


Figure 3: Distributions of Vina, QED, and SA scores over all the generated molecules.

**Metrics** We evaluate the quality of generated molecules from three main aspects: (1) **Binding Affinity** measures how well the generated molecules fit the binding site. We use Vina [33, 1] to compute the binding affinity (*Vina Score*). Before feeding the molecules to Vina, we employ the universal force fields (UFF) [26] to refine the generated structures following [21]. (2) **Drug Likeness** reflects how much a molecule is like a drug. We use *QED* score [4] as the metric for drug-likeness. (3) **Synthesizability** assesses the ease of synthesis of generated molecules. We use normalized *SA* score [6, 35] to measure molecules' synthesizability.

In order to evaluate the generation quality and diversity for each binding site, we define two additional metrics: (1) **Percentage of Samples with High Affinity**, which measures the percentage of a binding site's generated molecules whose binding affinity is *higher than or equal to* the reference ligand. (2) **Diversity** [16], which measures the diversity of generated molecules for a binding site. It is calculated by averaging pairwise Tanimoto similarities [3, 32] over Morgan fingerprints among the generated molecules of a target.

**Results** We first calculate Vina Score, QED, and SA for each of the generated molecules. Figure 3 presents the histogram of these three metrics and Table 1 shows the mean and median values of them over all generated molecules. For each binding site, we further calculate Percentage of Samples with High Affinity and Diversity. We report their mean and median values in the bottom half of Table 1. From the quantitative results, we find that in general, our model is able to discover diverse molecules that have higher binding affinity to specific targets. Besides, the generated molecules from our model also exhibit other desirable properties including fairly high drug-likeness and synthesizability. When compared to the CNN baseline liGAN [21], our method achieves clearly better performance on all metrics, especially on the drug-likeness score QED, which indicates that our model produces more realistic drug-like molecules.

To better understand the results, we select two binding sites in the testing set and visualize their top affinity samples for closer inspection (see the supplementary material for more cases). The two examples represent mediocre and better-than-average cases respectively. The top row of Figure 4 is the first example (PDB ID:`2hcj`). The average QED and SA scores of the generated molecules for this target are 0.483 and 0.663 respectively, around the median of these two scores. 8% of the generated molecules have higher binding affinity than the reference molecule, below the median 18.5%. The second example (PDB ID:`4rlu`) is shown in the bottom row. The average QED and SA scores are 0.728 and 0.785, and 18% of sampled molecules achieve higher binding affinity. From these two examples in Figure 4, we can see that the generated molecules have overall structures similar to the reference molecule and they share some common important substructures, which indicates that the generated molecules fit into the binding site as well as the reference one. Besides, the top affinity molecules generally achieve QED and SA score comparable to or even higher than the reference molecule, which reflects that the top affinity molecules not only fit well into the binding site but also exhibit desirable quality. In conclusion, the above two representative cases evidence the model's ability to generate drug-like and high binding affinity molecules for designated targets.

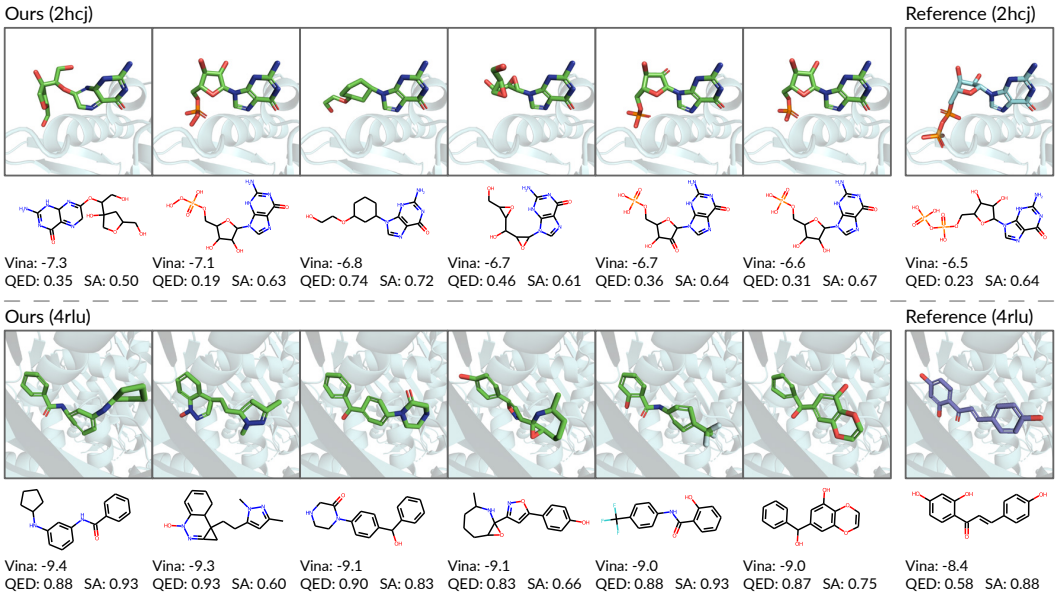


Figure 4: Generated molecules with top binding affinity and the reference molecule for two representative binding sites. Lower Vina score indicates higher binding affinity.

### 4.2 Linker Prediction

Linker prediction is to build a molecule that incorporates two given disconnected fragments in the context of a binding site [14]. Our model is capable of linker design without any task-specific adaptation or re-training. In specific, given a binding site and some fragments as input, we compose the initial context $\mathcal{C}_0$ containing both the binding site and the fragments. Then, we run the auto-regressive sampling algorithm to sequentially add atoms until the molecule is complete.

Table 2: Performance of linker prediction.

| Metric | | DeLinker | **Ours** |
|---|---|---|---|
| Similarity (↑) | Avg. | 0.612 | **0.701** |
| | Med. | 0.600 | **0.722** |
| Recovered (%, ↑) | | 40.00 | **48.33** |
| SC Score (%, ↑) | $\geq 0.7$ | 2.281 | 2.408 |
| | $\geq 0.8$ | 0.127 | 0.634 |
| Vina Score (kcal/mol, ↓) | Avg. | -8.512 | -8.603 |
| | Med. | -8.576 | -8.575 |

**Data Preparation** Following Imrie et al. [14], we construct fragments of each molecule in the testing set by enumerating possible double-cuts of acyclic single bonds. The pre-processing results in 120 data points in total. Each of them consists of two disconnected molecule fragments.

**Baselines** We compare our model with DeLinker [14]. Despite that DeLinker incorporates some 3D information, it is still a graph-based generative model. In contrast, our method operates fully in 3D space and thus is able to fully utilize the 3D context.

**Metrics** We assess the generated molecules from fragments with four main metrics: (1) **Similarity**: We use Tanimoto Similarity [32, 3] over Morgan fingerprints [16] to measure the similarity between the molecular graphs of generated molecule and the reference molecule. (2) **Percentage of Recovered Molecules**: We say a test molecule is recovered if the model is able to generate a molecule that perfectly matches it (Similarity = 1.0). We calculate the percentage of test molecules that are recovered by the model. (3) **SC Score**: Following Imrie et al. [14], we measure the shape-color similarity (SC) between generated molecules and the reference molecule. Specifically, we first dock the generated molecule to the binding site to obtain binding poses. Then we calculate the SC score based on the docking poses and the reference molecule. (4) **Binding Affinity**: We use Vina [1, 33] to compute the the generated molecules' binding affinity to the target.

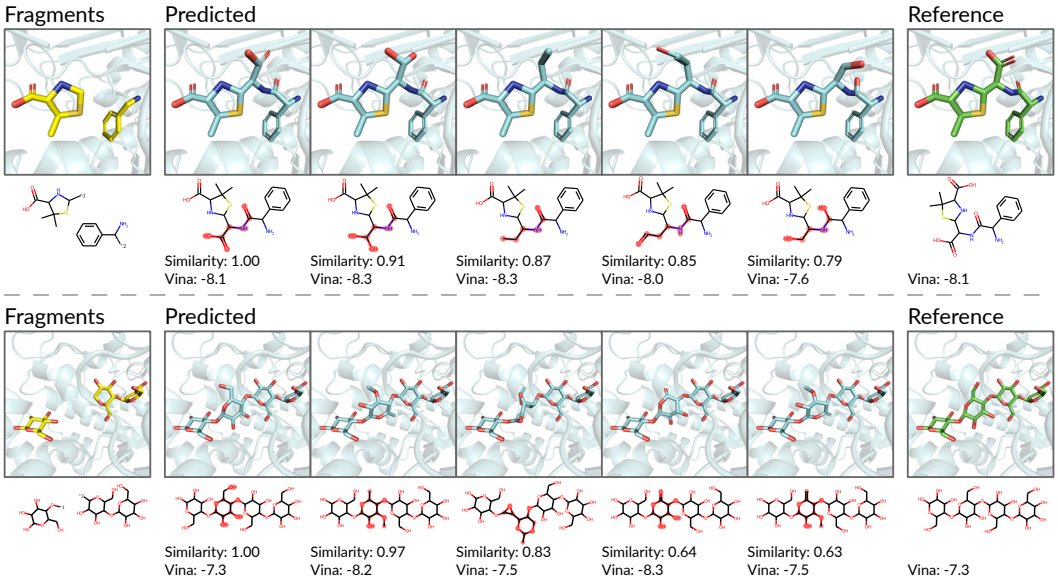


Figure 5: Two example of linker prediction. Atoms highlighted in red are predicted linkers.

**Results** For each data point, we use our model and DeLinker to generate 100 molecules. We first calculate the average similarity for each data point and report their overall mean and median values. Then, we calculate the percentage of test molecules that are successfully recovered by the model. We also calculate the percentage of generated molecules that achieve SC Score greater than 0.7 and 0.8, following Imrie et al., and report the mean and median percentage over the testing set. Finally, we use Vina to evaluate the generated molecules' binding affinity. These results are summarized in Table 2. As shown in the table, when measured by SC score and Vina score, our proposed method's performance is on par with the graph-based baseline DeLinker. However, our method clearly outperforms DeLinker on Similarity and Percentage of Recovery, suggesting that our method is able to link fragments in a more realistic way. In addition, we present two examples along with 5 generated molecules at different similarities in Figure 5. The example demonstrates the model's ability to generate suitable linkers. More examples are provided in the supplementary material.

## 5 Conclusions

In this paper, we propose a novel approach to structure-based drug design. In specific, we design a 3D generative model that estimates the probability density of atom's occurrences in 3D space and formulate an auto-regressive sampling algorithm. Combined with the sampling algorithm, the model is able to generate high-quality drug-like molecules for specific binding sites. By conducting extensive experiments, we demonstrate our model's effectiveness in designing molecules for specific targets. Our future work would include extending the model so that it can estimate the probability of fragments in addition to atoms, allowing fragment-based generation.

## References

[1] Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. Fast, accurate, and reliable molecular docking with quickvina 2. *Bioinformatics*, 31(13):2214–2216, 2015.

[2] Amy C. Anderson. The process of structure-based drug design. *Chemistry & Biology*, 10(9): 787–797, 2003. ISSN 1074-5521. doi: https://doi.org/10.1016/j.chembiol.2003.09.002. URL `https://www.sciencedirect.com/science/article/pii/S1074552103001947`.

[3] Dávid Bajusz, Anita Rácz, and Károly Héberger. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):1–13, 2015.

[4] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

[5] Esben Jannik Bjerrum and Richard Threlfall. Molecular generation with recurrent neural networks (rnns). *arXiv preprint arXiv:1705.04612*, 2017.

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

[7] 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.

[8] Niklas WA Gebauer, Michael Gastegger, and Kristof T Schütt. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. *arXiv preprint arXiv:1906.00957*, 2019.

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pages 1263–1272. PMLR, 2017.

[10] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[11] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

[12] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 4(2):268–276, 2018. ISSN 2374-7943. doi: 10.1021/acscentsci.7b00572.

[13] Paul CD Hawkins. Conformation generation: the state of the art. *Journal of Chemical Information and Modeling*, 57(8):1747–1756, 2017.

[14] Fergus Imrie, Anthony R Bradley, Mihaela van der Schaar, and Charlotte M Deane. Deep generative models for 3d linker design. *Journal of chemical information and modeling*, 60(4): 1983–1995, 2020.

[15] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2323–2332. PMLR, 2018.

[16] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Composing molecules with multiple property constraints. *arXiv preprint arXiv:2002.03244*, 2020.

[17] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[18] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR, 2017.

[19] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[20] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. Constrained graph variational autoencoders for molecule design. *arXiv preprint arXiv:1805.09076*, 2018.

[21] Tomohide Masuda, Matthew Ragoza, and David Ryan Koes. Generating 3d molecular structures conditional on a receptor binding site with deep generative models. *arXiv preprint arXiv:2010.14442*, 2020.

[22] AkshatKumar Nigam, Pascal Friederich, Mario Krenn, and Alan Aspuru-Guzik. Augmenting genetic algorithms with deep neural networks for exploring the chemical space. *arXiv preprint arXiv:1909.11655*, 2019.

[23] Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 3 (1):1–14, 2011.

[24] Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8):675–679, 2013.

[25] Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Learning a continuous representation of 3d molecular structures with deep generative models. *arXiv preprint arXiv:2010.08687*, 2020.

[26] Anthony K Rappé, Carla J Casewit, KS Colwell, William A Goddard III, and W Mason Skiff. Uff, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *Journal of the American chemical society*, 114(25):10024–10035, 1992.

[27] Kristof T Schütt, PJ Kindermans, Huziel E Sauceda, Stefan Chmiela, Alexandre Tkatchenko, and Klaus R Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA*, pages 1–11, 2017.

[28] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1): 120–131, 2018.

[29] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

[30] Gregor Simm, Robert Pinsler, and José Miguel Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. In *International Conference on Machine Learning*, pages 8959–8969. PMLR, 2020.

[31] Gregor NC Simm, Robert Pinsler, Gábor Csányi, and José Miguel Hernández-Lobato. Symmetry-aware actor-critic for 3d molecular design. *arXiv preprint arXiv:2011.12747*, 2020.

[32] Taffee T Tanimoto. Elementary mathematical theory of classification and prediction. 1958.

[33] 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.

[34] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28 (1):31–36, 1988.

[35] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.

## 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? [N/A]
   (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)? [No] We will provide the complete codes and data once this paper is published.
   (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? [N/A]
   (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? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]