Pocket2Mol: Efficient Molecular Sampling Based on 3D Protein Pockets

Xingang Peng [1]   **Shitong Luo** [2]   **Jiaqi Guan** [3]   **Qi Xie** [4]   **Jian Peng** [2 3 5]   **Jianzhu Ma** [2 6 7]

Abstract

Deep generative models have achieved tremendous success in designing novel drug molecules in recent years. A new thread of works have shown the great potential in advancing the specificity and success rate of *in silico* drug design by considering the structure of protein pockets. This setting posts fundamental computational challenges in sampling new chemical compounds that could satisfy multiple geometrical constraints imposed by pockets. Previous sampling algorithms either sample in the graph space or only consider the 3D coordinates of atoms while ignoring other detailed chemical structures such as bond types and functional groups. To address the challenge, we develop Pocket2Mol, an E(3)-equivariant generative network composed of two modules: 1) a new graph neural network capturing both spatial and bonding relationships between atoms of the binding pockets and 2) a new efficient algorithm which samples new drug candidates conditioned on the pocket representations from a tractable distribution without relying on MCMC. Experimental results demonstrate that molecules sampled from Pocket2Mol achieve significantly better binding affinity and other drug properties such as druglikeness and synthetic accessibility.

1. Introduction

Deep learning has achieved great success in drug design (Polykovskiy et al., 2020). One of the representative works is that two out of six compounds designed by Insilico

---

[1]Tsinghua University, Beijing, China [2]HeliXon Limited, Beijing, China [3]Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, USA [4]Westlake University, Hangzhou, China [5]AIR, Tsinghua University, Beijing, China [6]Institute for Artificial Intelligence, Peking University, Beijing, China [7]Beijing Institute for General Artificial Intelligence, Beijing, China. Correspondence to: Jianzhu Ma <majianzhu@pku.edu.cn>.

Medicine have been found to be active in cellular assays against kinase DDR1 (Zhavoronkov et al., 2019). The main intuition of these generative models is to efficiently represent all the collected chemical structures in a compact low dimensional space and sample new drug candidates by perturbing the hidden values. The outputs of these models vary from 1D chemical descriptors (e.g. SMILES strings (Gómez-Bombarelli et al., 2018)), 2D graphs (e.g. trees (Jin et al., 2018; De Cao & Kipf, 2018) and hyper-graphs (Kajino, 2019)) to 3D structures (Gebauer et al., 2018; 2019; Simm et al., 2020).

However, at the molecular level, a small molecule inhibits or activates particular biological functions only by binding to a specific protein pocket. It can be expected that drug design ignoring the target protein pockets could significantly reduce the success rate of downstream cell experiments. Only recently, an increasing amounts of works start to study the pocket-based drug design. To be more specific, given the 3D binding pocket of the target protein, these models are aware of the geometric information of the 3D pocket and generate molecules to bind to the pockets accordingly. Early approaches modify the pocket-free models by integrating evaluation functions like docking scores between sampled molecules and pockets to guide the candidate searching (Li et al., 2021). Another types of models transform the 3D pocket structures to molecular SMILES strings or 2D molecular graph (Skalic et al., 2019; Xu et al., 2021) without modeling the interactions between the small molecular structures and 3D pockets explicitly. Conditional generative models are developed to model the 3D atomic density distributions within the 3D pocket structures (Masuda et al., 2020; Luo et al., 2021). The challenging point of this problem then moves to the efficiency of the structural sampling algorithm from the learned distributions. In addition, previous models over-emphasize the importance of the 3D locations of atoms while ignoring the generation of chemical bonds, which results in unrealistic atomic connections in practice.

In this work, we improve the pocket-based drug design from the following directions: 1) developing a new deep geometric neural network to accurately model the 3D structures of pockets; 2) designing a new sampling strategy to enable more efficient conditional 3D coordinates sampling; 3) assigning the ability of sampling chemical bonds between a pair of atoms to our model. In particular, we develop

an E(3)-equivariant generative model, named Pocket2Mol, leveraging both vector-based neurons and geometric vector perceptrons (Deng et al., 2021; Jing et al., 2021) as basic building blocks for learning the chemical and geometric constraints imposed by protein pockets. Our model jointly predicts frontier atoms, atomic positions, atom types and chemical bonds with shared atom-level embedding and samples molecules in an auto-regressive way. Thanks to the vector-based neurons, the model can directly generate tractable distributions of the relative atomic coordinates relative to the focal atoms to avoid using the conventional MCMC algorithms (Luo et al., 2021). Experiment results demonstrate that drug candidates sampled by Pocket2Mol not only show higher binding affinities and drug-likeness but also contain more realistic sub-structures than the state-of-the-art models. In addition, Pocket2Mol is much faster than previous auto-regressive sampling algorithms based on MCMC.

To summarize, our key contributions include:

- We construct a new E(3)-equivariant neural network to capture the chemical and geometric constraints imposed by the 3D pockets.
- We propose an efficient conditional molecular sampling algorithm , characterizing a novel position generation strategy and accurate prediction of element types and chemical bonds.
- Drug-likeness is not enough to quantify whether a drug candidate is realistic in biology. We propose a new way to measure whether a unique sub-structure is learned from data or caused by algorithmic bias.
- Our method could sample drug candidates with much better properties than those of the state-of-the-art methods.

2. Related Work

Molecular Generation based on 3D Protein Pockets Here we only review the molecular generation approaches involving protein pockets. Skalic et al. (2019) employed a modified GAN model, named BicycleGAN (Zhu et al., 2017), to represent molecules in the hidden space from protein pockets and decode the representations to SMILES strings using a captioning network. Another representative work by Xu et al. (2021) designed two structure descriptors to encode the pocket and generated SMILES using conditional RNN. However, these methods merely generate 1D SMILES strings or 2D molecular graphs. Although they considered the 3D structures of pockets, it is not clear whether the generated compounds based on the form of SMILES strings or graphs could actually fit the geometric landscape of pockets ignoring the 3D structures of compounds.

Another thread of works starts to consider the 3D molecular structures of both pockets and small molecules. Li et al.

(2021) proposed a ligand neural network to generate 3D molecular structures and leverage Monte Carlo Tree Search to optimize candidate molecules binding to a specific pocket. Instead of involving 3D pockets in the training process, the function of 3D pockets is to evaluate the quality of drug candidates and guide the generation process.

Recently, multiple end-to-end learning models are proposed to directly translate 3D pockets to 3D molecular structures with high specificity. Masuda et al. (2020) proposed a conditional variational autoencoder to encode the 3D pockets and generate the voxelized atomic densities for sampling the drug candidates. The model was constructed based on the 3D CNN model which is not equivariant to rotation and hard to scale to large protein pockets. Luo et al. (2021) designed another generative model to estimate the probability density of atom types. The major improvement of this model over previous methods is to choose graph neural networks (GNN) as the backbone and introduce the mask-fill schema in the training process, which enhances the ability to estimate the rough landscape of the conditional probability. However, the model also has its own limitations: 1) it models the atomic densities while ignoring the chemical bonds, which often leads to unrealistic molecular structures; 2) the sampling algorithm relies on MCMC to explore the 3D space, which is sub-optimal and inefficient.

Vector Feature-based Equivariant Networks Another important technique related to our work is the equivariant neural network (Cohen & Welling, 2016; Schütt et al., 2017a; Weiler et al., 2018; Thomas et al., 2018; Anderson et al., 2019; Fuchs et al., 2020). It is common to adopt GNN-based architectures to achieve the global rotation equivariance for 3D objects (Klicpera et al., 2020; Satorras et al., 2021). However, they require the input and hidden features for each layer to be equivariant, which does not fit the vector features such as side-chain angles of each amino acid. To address this problem, Deng et al. (2021) proposes vector neurons which extends 1D scalar neurons to 3D vectors and also defines a set of equivariant operations in the vector space. Later, Jing et al. (2021) develops the geometric vector perceptrons to efficiently propagate information between scalar features and vector features. Schütt et al. (2021) introduces equivariant message passing for vector representations using equivariant features. However, they can only linearly combine 3D vectors to satisfy the rotation equivariance, in essence, limiting the model's geometric expressiveness.

Generation of Atom Positions The prediction of the atom positions is the key issue in the auto-regressive 3D molecular generation. A common strategy is to predict the distributions of distances between the new atoms and all previous atoms (Gebauer et al., 2018; 2019) and sample from the joint

distributions. However, the long range distances are hard to predict, which introduces additional errors in the atom generation. Another strategy is to build a local spherical coordinate systems and predict the positions in the local space (Simm et al., 2020), but the transformation between Euclidean space and spherical space is inefficient and not straightforward.

3. Method

The central idea of Pocket2Mol is to learn a probability distribution of atom or bond type for each location inside the pocket based on the atoms that already exist. To learn this context-specific distribution, we adopt the auto-regression strategy to predict the randomly masked part of the training drug from the rest of it.

In this section, we first introduce the designed generation procedure in Sec. 3.1 to elicit the modules our model will include. In Sec. 3.2, we will describe our graph neural network architecture in details. Specifically, we will show how our model achieve E(3)-equivariance leveraging recent advances and how the encoder module and predictor modules are designed. Finally, in Sec. 3.3, we will present the training objectives, by which the model learns which atoms are frontiers, where the new atom should be placed and which element type and bond types should be assigned.

3.1. Generation Procedure

Formally, the protein pocket is represented as a set of atoms with coordinates $\mathcal{P}^{(\text{pro})} = \{(a_i^{(\text{pro})}, \mathbf{r}_i^{(\text{pro})})\}_{i=1}^N$, where $a_i^{(\text{pro})}$ and $\mathbf{r}_i^{(\text{pro})}$ are the ith heavy atom identity and its coordinate respectively, and N is the number of atoms of the protein pocket. The molecules are sampled in a continuous way. The already generated molecular fragments with n atoms are denoted as a graph with coordinates $\mathcal{G}_n^{(mol)} = \{(a_i^{(\text{mol})}, \mathbf{r}_i^{(\text{mol})}, \mathbf{b}_i^{(\text{mol})})\}_{i=1}^n$ where $a_i^{(\text{mol})}$, $\mathbf{r}_i^{(\text{mol})}$ and $\mathbf{b}_i^{(\text{mol})}$ represent the ith heavy atom, its coordinate and its valence bonds with other atoms, respectively. The model is denoted as ϕ and the generation process is defined as follows:

$$\begin{aligned} \mathcal{G}_n^{(\text{mol})} &= \phi(\mathcal{G}_{n-1}^{(\text{mol})}, \mathcal{P}^{(\text{pro})}), \quad n > 1 \\ \mathcal{G}_1^{(\text{mol})} &= \phi(\mathcal{P}^{(\text{pro})}), \end{aligned} \quad (1)$$

For each atom, the generation procedure consists of four major steps (Fig. 1). First, the model's frontier predictor f_{fro} will predict the frontier atoms of the current molecular fragments. The frontiers are defined as molecular atoms that can covalently connect to new atoms. If all atoms are not frontiers, it indicates the current molecule is complete and the generation process terminates. Second, the model samples an atom from the frontier set as the focal atom.
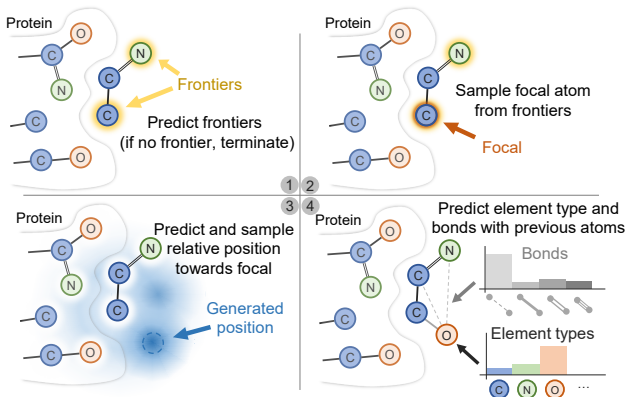

Figure 1. The Generation Procedure of Pocket2Mol. For each panel, the left part is protein and the right part is the sampled molecular fragment.

Third, based on the focal atom, the model's position predictor f_{pos} predicts the relative position of the new atom. In the end, the model's atom element predictor f_{ele} and bond type predictor f_{bond} will predict the probabilities of the element types and the bond types with existing atoms and then sample an element type and valence bonds for the new atom. In this way, the new atom is successfully added into the current molecular fragments and the generation process continues until no frontier atom could be found. Note that this generation process is different for the first atom since there is no molecular atom to be chosen as the frontier yet. For the first atom, all atoms of the protein pocket are used to predict the frontiers and here the frontiers are defined as atoms that the new atoms can be generated within 4 Å.

3.2. Model Architecture

Based on the generation procedure mentioned above, the model needs to be composed of four modules: an encoder, a frontier predictor, a position predictor and an element-and-bond predictor. Before we illustrate how each component is designed separately, we will first introduce the E(3)-Equivariant Neural Network (Sec. 3.2.1), which we applied as the main architecture to better capture the 3D geometric attributes of pockets and molecular fragments. Following the introduced equivariant network, we will present how the encoder module (Sec. 3.2.2) and predictor modules (Sec. 3.2.3) are constructed.

3.2.1. E(3)-EQUIVARIANT NEURAL NETWORK

It has been shown that representing the vertices and edges in a 3D graph with both scalar and vector features (Satorras et al., 2021; Schütt et al., 2021) can help boost the expressive power of the neural network. In our network, all vertices and edges in protein pockets $\mathcal{P}^{(\text{pro})}$ and molecular fragments $\mathcal{G}_n^{(mol)}$ are associated with both scalar and vector features

to better capture the 3D geometric information. In the rest of texts, we use symbols "$\cdot$" and "$\rightarrow$" overhead to explicitly denote scalar features and vector features (e.g. $\dot{\mathbf{x}}$ and $\vec{\mathbf{x}}$).

We adopt the geometric vector perceptrons (Jing et al., 2021) and the vector-based neural network (Deng et al., 2021) to achieve E(3)-equivariance. The geometric vector perceptrons (GVP) extend standard dense layers and can propagate information between scalar features and vector features (Jing et al., 2021). We briefly formulize the original GVP in the Appendix A for readers who are not familiar with it. The vector neuron networks expand a set of vanilla neural operations (e.g. linear layer, activation functions) to the vector feature space (Deng et al., 2021). In particular, we modify the original GVP by adding a vector nonlinear activation to the output vector of the GVP, which is denoted as G_{per}:

$$\begin{aligned} (\dot{\mathbf{x}}', \vec{\mathbf{x}}') &= \mathrm{GVP}(\dot{\mathbf{x}}, \vec{\mathbf{x}}) \\ \vec{\mathbf{x}}' &\leftarrow \mathrm{VectorNonLinear}(\vec{\mathbf{x}}') \end{aligned} \quad (2)$$

where $(\dot{\mathbf{x}}, \vec{\mathbf{x}})$ could be any pair of scalar-vector features. In our model, we choose the LeakyReLU nonlinear functions for both scalar and vector outputs of GVP. In addition, we define a geometric vector linear (GVL) block denoted as G_{lin} by dropping the nonlinear activations of both scalar and vector outputs of GVP. The modified GVP blocks G_{per} and the GVL blocks G_{lin} are the primary building blocks of our model, which enables the model to be E(3)-equivariant. Besides, including vector features is crucial for our model to directly and accurately predict the atom positions based on the geometric environments provided by the protein pockets (details will be described in Sec. 3.2.3).

3.2.2. ENCODER

We represent the protein pockets and molecular fragments as a k-nearest neighbor (KNN) graph in which vertices are atoms and each atom is connected to its k-nearest neighbors. The input scalar features of protein atoms include the element types, the amino acids they belong to and whether they are backbone or side-chain atoms. The input scalar features of molecular atoms include the element types, the valence and the numbers of different chemical bonds. In addition, all atoms have one more scalar feature indicating whether they belong to the protein or the molecular fragment. The scalar edge features include the distances encoded with Gaussian RBF kernels (Schütt et al., 2017b), the bond types and a boolean variable indicating whether there is a valence on the edge. The input vector vertex features include the coordinates of atoms, while the vector edge features are the unit directional vector of the edge in the 3D space.

First, multiple embedding layers are applied to embed these features as $(\dot{\mathbf{v}}_i^{(0)}, \vec{\mathbf{v}}_i^{(0)})$ for vertices and $(\dot{\mathbf{e}}_{ij}^{(0)}, \vec{\mathbf{e}}_{ij}^{(0)})$ for edges. Then L message passing modules $M_l(l = 1, \ldots, L)$ and update modules $U_l(l = 1, \ldots, L)$ are concatenated

interleavedly to learn the local structure representations:

$$\begin{aligned} (\dot{\mathbf{m}}_i^{(l)}, \vec{\mathbf{m}}_i^{(l)}) &= \sum_{j \in \mathrm{KNN}(i)} M_l(\dot{\mathbf{v}}_j^{(l-1)}, \vec{\mathbf{v}}_j^{(l-1)}, \dot{\mathbf{e}}_{ij}^{(l-1)}, \vec{\mathbf{e}}_{ij}^{(l-1)}) \\ (\dot{\mathbf{v}}_i^{(l)}, \vec{\mathbf{v}}_i^{(l)}) &= U_l(\dot{\mathbf{v}}_i^{(l-1)}, \vec{\mathbf{v}}_i^{(l-1)}, \dot{\mathbf{m}}_i^{(l)}, \vec{\mathbf{m}}_i^{(l)}) \end{aligned} \quad (3)$$

The message passing module is formulated as:

$$\begin{aligned} (\dot{\mathbf{v}}_j', \vec{\mathbf{v}}_j') &= G_{\mathrm{lin}}(\dot{\mathbf{v}}_j^{(l-1)}, \vec{\mathbf{v}}_j^{(l-1)}) \\ (\dot{\mathbf{e}}_{ij}', \vec{\mathbf{e}}_{ij}') &= G_{\mathrm{per}}(\dot{\mathbf{e}}_{ij}^{(l-1)}, \vec{\mathbf{e}}_{ij}^{(l-1)}) \\ \dot{\mathbf{m}}_j' &= \dot{\mathbf{v}}_j' \circ \dot{F}_{\mathrm{lin}}(\dot{\mathbf{e}}_{ij}') \\ \vec{\mathbf{m}}_j' &= \dot{F}_{\mathrm{lin}}'(\dot{\mathbf{e}}_{ij}') \circ \vec{\mathbf{v}}_j' + \dot{F}_{\mathrm{lin}}''(\dot{\mathbf{v}}_j') \circ \vec{F}_{\mathrm{lin}}(\vec{\mathbf{e}}_{ij}') \\ (\dot{\mathbf{m}}_j^{(l)}, \vec{\mathbf{m}}_j^{(l)}) &= G_{\mathrm{per}}'(\dot{\mathbf{m}}_j', \vec{\mathbf{m}}_j') \end{aligned} \quad (4)$$

where $\circ$ stands for Hadamard product, $\dot{F}_{\mathrm{lin}}$ and $\vec{F}_{\mathrm{lin}}$ denote the scalar linear layer and the vector neuron linear layer respectively. The GVL block G_{lin} and GVP block G_{per} work like conventional linear layers and perceptrons to transform both scalar and vector features. The scalar parts of the messages $\dot{\mathbf{m}}_j'$ are the product of scalar features of both vertices and edges. However, the vector messages $\vec{\mathbf{m}}_j'$ cannot be the product of vector features of vertices and edges because this violates the equivariant constraints. Instead, the vector messages are calculated as the summation of vector features of vertices and edges after the multiplication with scalar features, which allows the information interaction between vertices and edges as well as between scalar and vector features. The update module is formulated as:

$$\begin{aligned} (\dot{\mathbf{v}}_i^{(l-1)\prime}, \vec{\mathbf{v}}_i^{(l-1)\prime}) &= G_{\mathrm{lin}}'(\dot{\mathbf{v}}_i^{(l-1)}, \vec{\mathbf{v}}_i^{(l-1)}) \\ \dot{\mathbf{v}}_i^{(l)} &= \dot{\mathbf{v}}_i^{(l-1)\prime} + \dot{\mathbf{m}}_i^{(l)} \\ \vec{\mathbf{v}}_i^{(l)} &= \vec{\mathbf{v}}_i^{(l-1)\prime} + \vec{\mathbf{m}}_i^{(l)} \end{aligned} \quad (5)$$

which is a simple addition operation to the messages after going through the GVL block.

3.2.3. PREDICTORS

The encoder functions extract hidden representations $(\dot{\mathbf{v}}_i^{(L)}, \vec{\mathbf{v}}_i^{(L)})$ capturing the chemical and geometric attributes of the protein pockets and molecule fragments. These representations are used for frontier predictor, position predictor and element-and-bond predictor (we omit superscript L in the following for simplicity).

Frontier Predictor We define a geometric vector MLP (GV-MLP) as a GVP block followed by a GVL block, denoted as G_{mlp}. The frontier predictor takes the features of atom i as input and utilizes one GV-MLP layer to predict the probability of being a frontier p_{fro} as follows,

$$\begin{aligned} (\dot{p}_{\mathrm{fro}}', \vec{\mathbf{p}}_{\mathrm{fro}}') &= G_{\mathrm{mlp}}^{(\mathrm{fro})}(\dot{\mathbf{v}}_i, \vec{\mathbf{v}}_i) \\ p_{\mathrm{fro}} &= \sigma(\dot{p}_{\mathrm{fro}}') \end{aligned} \quad (6)$$

where σ is the sigmoid activation function.

Position Predictor   The position predictor takes as input the features of focal atom i and predict the relative position of the new atom. Since the vector features are equivariant in our model, they can be directly used to generate the relative coordinates $\Delta \mathbf{r}_i$ towards the focal atom coordinate $\mathbf{r}_i$. We build the output of position predictor as a Gaussian Mixture Model with diagonal covariance $p(\Delta \mathbf{r}_i) = \sum_{k=1}^{K} \pi_i^{(k)} \mathcal{N}(\boldsymbol{\mu}_i^{(k)}, \boldsymbol{\Sigma}_i^{(k)})$ in which parameters are predicted by multiple neural networks as follows:

$$(\dot{\mathbf{v}}_i^{(\text{pos})}, \vec{\mathbf{v}}_i^{(\text{pos})}) = G_{\text{mlp}}^{(\text{pos})}(\dot{\mathbf{v}}_i, \vec{\mathbf{v}}_i)$$

$$(\dot{\boldsymbol{\mu}}_i', \vec{\boldsymbol{\mu}}_i') = G_{\text{lin}}^{(\mu)}(\dot{\mathbf{v}}_i^{(\text{pos})}, \vec{\mathbf{v}}_i^{(\text{pos})})$$

$$(\dot{\boldsymbol{\Sigma}}_i', \vec{\boldsymbol{\Sigma}}_i') = G_{\text{lin}}^{(\Sigma)}(\dot{\mathbf{v}}_i^{(\text{pos})}, \vec{\mathbf{v}}_i^{(\text{pos})}) \qquad (7)$$

$$(\dot{\boldsymbol{\pi}}_i', \vec{\boldsymbol{\pi}}_i') = G_{\text{lin}}^{(\pi)}(\dot{\mathbf{v}}_i^{(\text{pos})}, \vec{\mathbf{v}}_i^{(\text{pos})})$$

$$(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \boldsymbol{\pi}_i) = (\vec{\boldsymbol{\mu}}_i', \exp(\vec{\boldsymbol{\Sigma}}_i'), \sigma_{(\text{sf})}(\dot{\boldsymbol{\pi}}_i'))$$

where $\sigma_{(\text{sf})}$ is the softmax function. After processed by GV-MLP, the mean, covariance and prior probability of Gaussian components are predicted by three separate GVL blocks. Since the vector features are equivariant, the vector outputs of GVL block can directly represent the mean vectors and covariance vectors.

Element-and-Bond Predictor   After predicting the position of the new atom i, the element-and-bond predictor will predict the element type of the new atom i and the valence bonds between atom i and all the atoms q ($\forall q \in \mathcal{V}^{(mol)}$) in the existing molecular fragment. Fig. 2 demonstrates the architecture of the predictor neural network. First, we collect k-nearest neighbor atoms $j \in \text{KNN}(i)$ among all atoms. Then a message passing module (Eq. 4) is utilized to integrate the local information from neighbor atoms into the new atom i position as its representation $(\dot{\mathbf{v}}_i, \vec{\mathbf{v}}_i)$, from which the element type of atom i is predicted.

In a parallel pathway, the representation of edge between atom i and q, denoted as $(\dot{\mathbf{z}}_{iq}, \vec{\mathbf{z}}_{iq})$, is the concatenation of the features of atom i, the features of atom q and the processed features of the edge e_{iq}, followed by a GV-MLP block, i.e.

$$\dot{\mathbf{z}}_{iq}^{(0)} = [\dot{\mathbf{v}}_i; \dot{\mathbf{v}}_q; \dot{\mathbf{e}}_{iq}']$$

$$\vec{\mathbf{z}}_{iq}^{(0)} = [\vec{\mathbf{v}}_i; \vec{\mathbf{v}}_q; \vec{\mathbf{e}}_{iq}'] \qquad (8)$$

$$(\dot{\mathbf{z}}_{iq}, \vec{\mathbf{z}}_{iq}) = G_{\text{mlp}}^{(\text{bon})}(\dot{\mathbf{z}}_{iq}^{(0)}, \vec{\mathbf{z}}_{iq}^{(0)})$$

where $(\dot{\mathbf{e}}_{iq}', \vec{\mathbf{e}}_{iq}')$ are the input edge features processed by the edge embedding and one GV-MLP block.

The edge representations are fed into an attention modules to predict the bond types (we regard *no bond* as a special
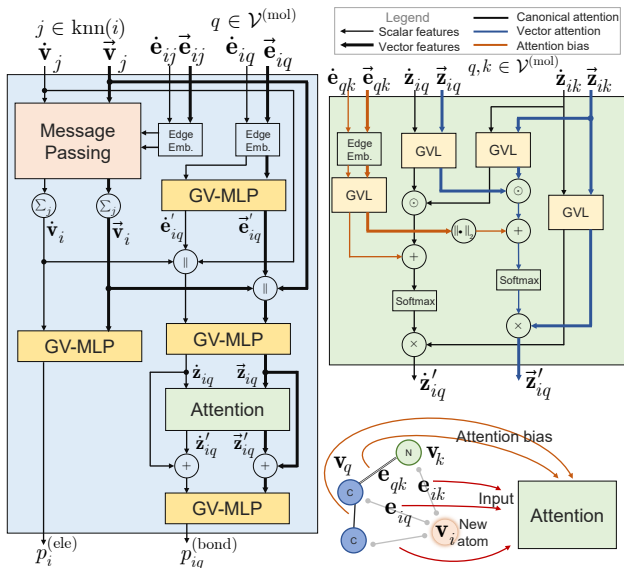

Figure 2. The element-and-bond predictor of the model and the detailed attention module for bond predictions. The symbol $\parallel$ stands for concatenation and the symbol $\odot$ stands for the inner product operation. The symbol $\parallel \cdot \parallel_2$ represents the norm of vectors.

bond type). As shown in Fig. 2, the attention module of Pocket2Mol differs from the conventional attention module in two main aspects: First, scalar and vector features processed by GVL blocks pass through two separate attention pathways. We choose the canonical attention (Vaswani et al., 2017) for scalar features. For the vector features, we propose a new type of attention module defined as follows:

$$a_{qk} = \langle \vec{\mathbf{z}}_{iq}^{(\text{att})}, \vec{\mathbf{z}}_{ik}^{(\text{att})} \rangle_F = \text{tr}\left(\vec{\mathbf{z}}_{iq}^{(\text{att})} \cdot \vec{\mathbf{z}}_{ik}^{(\text{att})\top} \right)$$

$$a_{qk}' = \sigma_{(\text{sf})}(a_{qv}) \qquad (9)$$

$$\vec{\mathbf{z}}_{iq}' = \sum_{v \in \mathcal{V}^{(\text{mol})}} a_{qv}' \vec{\mathbf{z}}_{iv}^{(\text{att})}$$

where $\vec{\mathbf{z}}_{iq}^{(\text{att})}, \vec{\mathbf{z}}_{ik}^{(\text{att})}$ and $\vec{\mathbf{z}}_{iv}^{(\text{att})}$ are the queries, keys and values of vector features, respectively. $\langle \, , \, \rangle_F$ is the Frobenius inner product. The basic rationale of vector-attention module is to generalize the dot product between queries and keys to Frobenius inner product. We prove this operation still keeps the equivariant property in Appendix C. Second, we add an attention bias to the product of queries and keys, which is inspired by the triangular self-attention module of AlphaFold (Jumper et al., 2021), where the attention bias is designed to capture the geometric constraints. The intuition here is shown in the right bottom of Fig. 2: the attention score of the edge $\mathbf{e}_{iq}$ to the edge $\mathbf{e}_{ik}$ is also related to the edge $\mathbf{e}_{qk}$. For example, if there is a double bond between atom q and k, it is nearly impossible that atom i can form valence bond with atom q and k simultaneously.

3.3. Training

In the training stage, we randomly mask atoms of molecules and train the model to recover the masked atoms. Specifically, for each pocket-ligand pair, we sample a mask ratio from the uniform distribution $U[0, 1]$ and mask corresponding number of molecular atoms. The remaining molecular atoms that have valence bonds to the masked atoms are defined as frontiers. Then the position predictor and the element-and-bond predictor try to recover the masked atoms that have valence bonds to the frontiers by predicting their positions towards corresponding frontiers, the element types and the bonds with remaining molecular atoms. If all molecular atoms are masked, the frontiers are defined as protein atoms that have masked atoms within 4Å and the masked atoms around the frontiers are to be recovered. For the element type prediction, similar with (Luo et al., 2021), we add one more element type representing *Nothing* at the query position. During the training process, we sample not only the positions of masked atoms for element type predictions but also negative positions from the ambient space and assign their labels as *Nothing*.

The loss of the frontier prediction, $\mathcal{L}_{\text{fro}}$, is the binary cross entropy loss of predicted frontiers. The loss of the position predictor, $\mathcal{L}_{\text{pos}}$, is the negative log likelihood of the masked atom positions. For the element type and bond type prediction, we used cross entropy losses for the classification, denoted as $\mathcal{L}_{\text{ele}}$ and $\mathcal{L}_{\text{bond}}$ respectively. The overall loss function is the summation of the above four loss functions:

$$\mathcal{L} = \mathcal{L}_{\text{fro}} + \mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{ele}} + \mathcal{L}_{\text{bond}} \quad (10)$$

We adopt Adam (Kingma & Ba, 2014) optimizer to optimize the encoder and all three predictors simultaneously.

4. Results

To evaluate the generation performance of Pocket2Mol, we use the CrossDocked dataset (Francoeur et al., 2020) which contains 22.5 million protein-molecule structures and follow the same data preparation and data splitting as (Luo et al., 2021) and (Masuda et al., 2020). In the test stage, we randomly sample 100 molecules for each protein pocket in the test set. We compare Pocket2Mol with two baseline methods: a CVAE-based model (CVAE) (Masuda et al., 2020) and another auto-regressive generative model (AR) (Luo et al., 2021). **The hyper-parameters of model architectures and training settings could be found in Appendix B.** The codes are available at `https://github.com/pengxingang/Pocket2Mol`.

Table 1. The comparison of general properties of the molecules in the test set and those generated by different methods.

	Test Set	CVAE	AR	Pocket2 Mol
Vina Score (kcal/mol, ↓)	-7.158 ± 2.10	-6.144 ± 1.57	-6.215 ± 1.54	**-7.288** ± 2.53
High Affinity (↑)	-	0.238 ± 0.28	0.267 ± 0.31	**0.542** ± 0.32
QED (↑)	0.484 ±0.21	0.369 ±0.22	0.502 ±0.17	**0.563** ±0.16
SA (↑)	0.732 ±0.14	0.590 ±0.15	0.675 ±0.14	**0.765** ±0.13
LogP	0.947 ±2.65	-0.140 ±2.73	0.257 ±2.01	**1.586** ±1.82
Lipinski (↑)	4.367 ±1.14	4.027 ±1.38	4.787 ±0.50	**4.902** ±0.42
Sim. Train (↓)	-	0.460 ±0.18	0.409 ±0.19	**0.376** ±0.22
Diversity (↑)	-	0.654 ±0.12	**0.742** ±0.09	0.688 ±0.14
Time (s, ↓)	-	-	19658.56 ±14704	**2503.51** ±2207

4.1. Evaluation of general properties of sampled molecules

In this section, we choose metrics that are widely adopted to evaluate the properties of the sampled drug candidates (Polykovskiy et al., 2020): (1) **Vina Score** estimates the binding affinity between the generated molecules and the protein pockets; (2) **High Affinity** is calculated as the percentage of pockets whose generated molecules have higher or equal affinities to the ones in the test set; (3) **QED** is the quantitative estimation of drug-likeness, a value measuring how likely a molecule is a potential candidate for a drug; (4) **SA** is the synthetic accessibility score, representing the difficulty of drug synthesis (normalized between 0 and 1 and higher values indicate easier synthesis); (5) **LogP** represents the octanol-water partition coefficient, and in general logP values should be between -0.4 and 5.6 to be good drug candidates (Ghose et al., 1998); (6) **Lipinski** measures how many rules the drug follow the Lipinski's rule of five (Lipinski et al., 1997; Veber et al., 2002), which is an empirical law of drug-likeness; (7) **Sim.Train** stands for the Tanimoto similarity (Bajusz et al., 2015) with the most similar molecules in the training set; (8) **Diversity** is calculated as the average Tanimoto similarities of the generated molecules for the individual pockets; (9) **Time** represents the time cost to generate 100 valid molecules for a pocket. The Vina Score is calculated by QVina (Alhossary et al., 2015) and the chemical properties are calculated by RDKit (Landrum, 2016).

The mean values of the above metrics are shown in Tab. 1. In general, Pocket2Mol outperforms the other two methods. An interesting observation is that the Vina scores of Pocket2Mol molecules are not only better than other computational models, but also better than the molecules in the test set, indicating that Pocket2Mol has the potential to generate molecules with better affinities with the pockets. In particular, Pocket2Mol successfully generates molecules better than the naturally existing molecules over half of the pockets, almost two-fold higher than the other two approaches. The drug potentials (QED, SA, LogP and Lipinski) are also significantly better than the others, which indicates that molecules generated by Pocket2Mol are more likely to be a drug candidate. In addition, molecules generated by Pocket2Mol have achieved lowest similarities than the molecules in the training set suggesting that Pocket2Mol does not merely memorize the training data. We also observe the diversity of Pocket2Mol is lower than another methods. The diversity metric is commonly adopted to quantify the quality of molecule generation without considering the constraints introduced by the protein pockets. For this pocket-based generation task, a better model capturing the geometric landscape of pockets should not necessarily achieve higher diversity because protein pocket is known to have strong specificity, which is also part of the reason why it often takes decades for biochemists to design one kind of drugs. For instance, for the well-known cancer target protein EGFR (epidermal growth factor receptor), the pocket shape is rough and unique and it is hard to believe that thousands of diverse molecules could fit this pocket (Robichaux et al., 2021) if considering the chemical and geometrical attributes of the pocket. Therefore, we do not think diversity is an important metric for this task as for many other drug generation tasks. In the end, the generation time of Pocket2Mol is around ten-fold lower than the AR model, which benefits from the direct generation of atom positions instead of randomly exploring the large 3D space using MCMC.

4.2. Sub-structure analysis

During evaluation, we find many approaches could achieve reasonable good performance on conventional metrics such as binding affinity and QED but the detailed sub-structures of their sampled molecules are unrealistic.

Therefore, we first visualize the generated molecules and find molecules produced by AR contain numerous distorted benzene rings and excessive three-atom rings (Fig. 3). The ring structures composed of three atoms are rare in the dataset (3%), and both previous approaches generate up to 30% of the three-atom ring substructures, which is obviously caused by algorithm bias. To systematically investigate this issue, we analyze the ratio of ring-structures with different sizes in the training set and the molecules generated by dif-
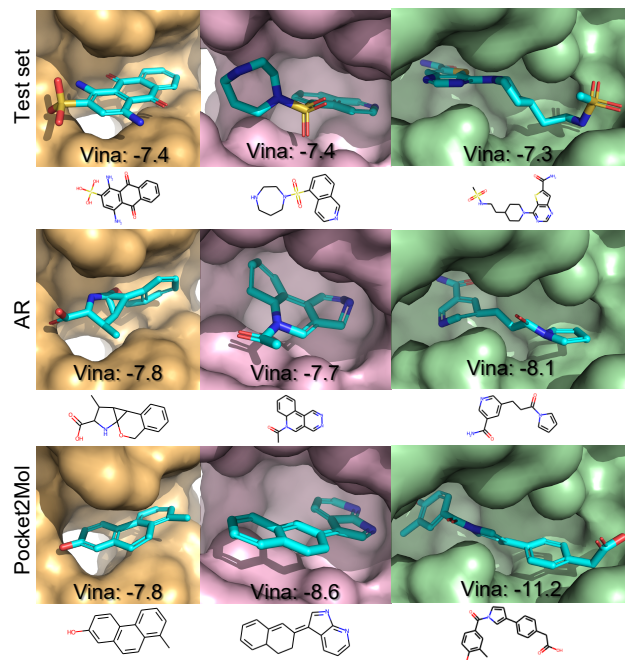

Figure 3. Three examples of generated molecules for pockets. The three columns represent three different pockets (PDB ids are 14GS, 5I0B and 5D7N respectively.)

ferent approaches. Our hypothesis is that even the chemical structures of the sampled molecules might be different from the test set but the distributions of basic function groups and sub-structures should be maintained, otherwise it would introduce additional difficulty for molecular biologists to understand the biological functions of these molecules.

As shown in Tab. 2, the molecules generated by Pocket2Mol show more similar ratios of rings of different sizes with the test set than the other two approaches. In particular, both CVAE and AR are inclined to generate extra amount of three-membered rings, which is caused by adding the chemical bonds after the generation of atoms using OpenBabel (O'Boyle et al., 2011). In contrast, Pocket2Mol produces more realistic chemical compounds by jointly learning atom and bond distributions and predicting bonds during the generation process. Another interesting observation is that, for the common five-membered and six-membered rings, Pocket2Mol also achieves most similar ratio to the training and test sets among all three models.

We further evaluate whether the distributions of the bond angles and dihedral angles of the sampled molecules agree with the test set by using the Kullback-Leibler (KL) divergence. We first find out common bond pairs and bond triples and then use RDKit to calculate the bond angles and the dihedral angles. We also include the KL divergence between the validation set and the test set for reference. As shown in Tab. 3, molecules generated by the Pocket2Mol show much

Table 2. The ratio of the molecules containing different rings in the datasets and those generated by different methods.

Ring Size	Train Set	Test Set	CVAE	AR	Pocket2 Mol
3	0.034	0.033	0.361	0.484	**0.002**
4	0.005	0.000	0.248	0.005	**0.000**
5	0.572	0.475	0.397	0.276	**0.415**
6	0.903	0.833	0.300	0.693	**0.885**
7	0.028	0.017	0.044	**0.033**	0.076
8	0.001	0.000	0.014	**0.007**	**0.007**
9	0.000	0.000	0.006	0.006	**0.002**

lower KL divergence than the other methods, indicating that the molecules generated by Pocket2Mol capture more geometric attributes of data.

Table 3. The KL divergence of the bond angles and dihedral angles with the test set. The lower letters represent the atoms in the aromatic rings.

	Val. Set	Test. Set	CVAE	AR	Pocket2 Mol
CCC	0.12	0.00	7.08	1.80	**0.97**
CCO	0.10	0.00	7.58	2.02	**0.95**
CNC	0.11	0.00	7.74	2.86	**0.49**
OPO	0.10	0.00	4.72	2.06	**0.23**
NCC	0.09	0.00	7.86	2.55	**0.95**
CC=O	0.07	0.00	7.41	2.90	**0.76**
COC	0.12	0.00	6.32	3.88	**0.24**
CCCC	0.14	0.00	**0.59**	0.78	0.71
cccc	0.08	0.00	7.91	10.64	**4.49**
CCCO	0.55	0.00	0.94	1.23	**0.56**
OCCO	1.01	0.00	1.92	1.85	**1.56**
Cccc	0.28	0.00	5.78	7.91	**2.85**
CC=CC	0.68	0.00	4.96	7.07	**4.09**

Moreover, to ensure the validity of generated 3D molecular conformers, we calculate the root-mean-square error (RMSD) between the generated molecular structures and the predicted structures by RDKit. Since a molecule can have multiple potential conformers, we sampled 20 conformers for each molecule using RDKit and choose the one with the minimum RMSD. The distributions of the RMSD of the generated molecules are shown in Fig. 4. The molecular structures generated by Pocket2Mol generally have lowest RMSD, indicating the correctness of the 3D molecular structures. As discussed in the previous section, accurately predicting the 3D structure of small molecules is crucial for studying their physical and chemistry properties (Liu et al., 2021), and more importantly, can determine whether they could actually bind to the 3D pocket. For instance, the algorithm samples a small molecule A with a particular

3D structure designed based on the 3D pocket. However, if A folds to another completely different 3D structure in the living cell, then it is expected the binding affinity of A and the pocket is low.
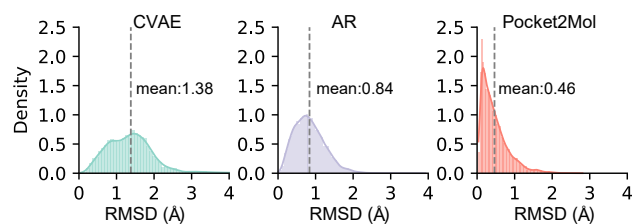

Figure 4. The distributions of RMSD of the generated 3D molecular structures.

5. Conclusion

Generation of 3D drug-like molecules for protein pockets is a significant but challenging task. In this work, we design Pocket2Mol, an E(3)-equivariant generative network composed of a graph neural network to model the chemical and geometrical features of the 3D protein pockets and a new efficient algorithm to sample new 3D drug candidates conditioned on the pockets. Experiments have demonstrated that molecules generated by Pocket2Mol not only show better affinities and chemical properties but also contain more realistic and accurate structures. We believe that Pocket2Mol will serve as a valuable tool for drug design.

Software

All the experiments are conducted on Ubuntu Linux with V100 GPUs. The codes are implemented in Python 3.8 mainly with Pytorch 1.9.0 and our codes are uploaded as Supplementary Material.

References

Alhossary, A., Handoko, S. D., Mu, Y., and Kwoh, C.-K. Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics*, 31(13):2214–2216, 02 2015. ISSN 1367-4803. doi: 10.1093/bioinformatics/btv082. URL https://doi.org/10.1093/bioinformatics/btv082.

Anderson, B., Hy, T.-S., and Kondor, R. Cormorant: Covariant molecular neural networks. *arXiv preprint arXiv:1906.04015*, 2019.

Bajusz, D., Rácz, A., and Héberger, K. Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? *Journal of cheminformatics*, 7(1):1–13, 2015.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

De Cao, N. and Kipf, T. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., and Guibas, L. Vector neurons: A general framework for so (3)-equivariant networks. *arXiv preprint arXiv:2104.12229*, 2021.

Francoeur, P. G., Masuda, T., Sunseri, J., Jia, A., Iovanisci, R. B., Snyder, I., and Koes, D. R. 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.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020.

Gebauer, N. W., Gastegger, M., and Schütt, K. T. Generating equilibrium molecules with deep neural networks. *arXiv preprint arXiv:1810.11347*, 2018.

Gebauer, N. W., Gastegger, M., and Schütt, K. T. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. *arXiv preprint arXiv:1906.00957*, 2019.

Ghose, A. K., Viswanadhan, V. N., and Wendoloski, J. J. A knowledge-based approach in designing combinatorial or medicinal chemistry libraries for drug discovery. 1. a qualitative and quantitative characterization of known drug databases. *Journal of Combinatorial Chemistry*, 1 (1):55–68, December 1998. doi: 10.1021/cc9800071. URL https://doi.org/10.1021/cc9800071.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J., and Dror, R. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

Jing, B., Eismann, S., Soni, P. N., and Dror, R. O. Equivariant graph neural networks for 3d macromolecular structure, 2021.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kajino, H. Molecular hypergraph grammar with its application to molecular optimization. In *International Conference on Machine Learning*, pp. 3183–3191. PMLR, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

Landrum, G. Rdkit: Open-source cheminformatics software. 2016. URL https://github.com/rdkit/rdkit/releases/tag/Release_2016_09_4.

Li, Y., Pei, J., and Lai, L. Structure-based de novo drug design using 3d deep generative models. *Chemical science*, 12(41):13664–13675, 2021.

Lipinski, C. A., Lombardo, F., Dominy, B. W., and Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews*, 23(1-3):3–25, 1997.

Liu, S., Wang, H., Liu, W., Lasenby, J., Guo, H., and Tang, J. Pre-training molecular graph representation with 3d geometry. *arXiv preprint arXiv:2110.07728*, 2021.

Luo, S., Guan, J., Ma, J., and Peng, J. A 3d generative model for structure-based drug design. *Advances in Neural Information Processing Systems*, 34, 2021.

Masuda, T., Ragoza, M., and Koes, D. R. Generating 3d molecular structures conditional on a receptor binding site with deep generative models. *arXiv preprint arXiv:2010.14442*, 2020.

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., and Hutchison, G. R. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 3(1): 1–14, 2011.

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., et al. Molecular sets (moses): a benchmarking platform for molecular

generation models. *Frontiers in pharmacology*, 11:1931, 2020.

Robichaux, J. P., Le, X., Vijayan, R., Hicks, J. K., Heeke, S., Elamin, Y. Y., Lin, H. Y., Udagawa, H., Skoulidis, F., Tran, H., et al. Structure-based classification predicts drug response in egfr-mutant nsclc. *Nature*, 597(7878): 732–737, 2021.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021.

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., and Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8 (1):1–8, 2017a.

Schütt, K. T., Kindermans, P.-J., Sauceda, H. E., Chmiela, S., Tkatchenko, A., and Müller, K.-R. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *arXiv preprint arXiv:1706.08566*, 2017b.

Schütt, K. T., Unke, O. T., and Gastegger, M. Equivariant message passing for the prediction of tensorial properties and molecular spectra. *arXiv preprint arXiv:2102.03150*, 2021.

Simm, G. N., Pinsler, R., Csányi, G., and Hernández-Lobato, J. M. Symmetry-aware actor-critic for 3d molecular design. *arXiv preprint arXiv:2011.12747*, 2020.

Skalic, M., Sabbadin, D., Sattarov, B., Sciabola, S., and De Fabritiis, G. From target to drug: Generative modeling for the multimodal structure-based ligand design. *Molecular Pharmaceutics*, 16(10):4282–4291, 2019. doi: 10.1021/acs.molpharmaceut.9b00634. PMID: 31437001.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Veber, D. F., Johnson, S. R., Cheng, H.-Y., Smith, B. R., Ward, K. W., and Kopple, K. D. Molecular properties that influence the oral bioavailability of drug candidates. *Journal of Medicinal Chemistry*, 45(12):2615–2623, May 2002. doi: 10.1021/jm020017n. URL https://doi.org/10.1021/jm020017n.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

Xu, M., Ran, T., and Chen, H. De novo molecule design through the molecular generative model conditioned by 3d information of protein binding sites. *Journal of Chemical Information and Modeling*, 61(7):3240–3254, 2021. doi: 10.1021/acs.jcim.0c01494. PMID: 34197105.

Zhavoronkov, A., Ivanenkov, Y. A., Aliper, A., Veselov, M. S., Aladinskiy, V. A., Aladinskaya, A. V., Terentiev, V. A., Polykovskiy, D. A., Kuznetsov, M. D., Asadulaev, A., et al. Deep learning enables rapid identification of potent ddr1 kinase inhibitors. *Nature biotechnology*, 37 (9):1038–1040, 2019.

Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., and Shechtman, E. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, 2017.

A. The formula of the original geometric vector perceptron

Here we briefly formulize the original geometric geometric vector perceptron (GVP) and the details can be found in Jing et al. (2020). The input and output of GVP are both scalar-vector pairs, denoted as $(\dot{\mathbf{x}}', \vec{\mathbf{x}}') = \mathrm{GVP}(\dot{\mathbf{x}}, \vec{\mathbf{x}})$. The original GVP is defined as:

$$
\begin{aligned}
\vec{\mathbf{x}}_1 &= \mathbf{W}_1 \vec{\mathbf{x}} \in \mathbb{R}^{h_1 \times 3} \\
\vec{\mathbf{x}}_2 &= \mathbf{W}_2 \vec{\mathbf{x}_1} \in \mathbb{R}^{h_2 \times 3} \\
\dot{\mathbf{s}} &= \|\vec{\mathbf{x}_1}\|_2 \in \mathbb{R}^{h_1} \\
\dot{\mathbf{x}}_1 &= \mathbf{W}_3 \begin{bmatrix} \dot{\mathbf{s}} \\ \dot{\mathbf{x}} \end{bmatrix} + \mathbf{b}_3 \in \mathbb{R}^{h_3} \\
\dot{\mathbf{x}}' &= \sigma_{\mathrm{act}}(\dot{\mathbf{x}}_1) \\
\vec{\mathbf{x}}' &= \sigma(\mathbf{W}_g \dot{\mathbf{x}}_1 + \mathbf{b}) \circ \vec{\mathbf{x}}_2
\end{aligned}
\tag{11}
$$

where $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters, $\| \cdot \|$ is row-wise norm and $\circ$ is Hadamard product. σ is the sigmoid activation function and σ_{act} is arbitrary activation function which is chosen as Leaky ReLU in our model. We also omit the bias terms in the original GVP.

B. The hyper-parameters of Pocket2Mol and training

The number of message passing layers L are 6. In the encoder, the hidden dimensions of vertices are fixed as 256 and 64 for scalar features and vector features, respectively. The embedding dimensions of edges are 64 for both scalar and vector features. In the frontier predictor, the hidden dimensions are 128 and 32 for scalar features and vector features, respectively. In the position predictors, the hidden dimensions are both 128 for scalar and vector features because the vector features are important in position prediction. The number of components in the Gaussian Mixture Model is 3 because we found most atoms in the dataset had no more than three connected atoms. In the element-and-bond predictor, the hidden dimensions are 128 and 32 for scalar features and vector features, respectively. The embedding dimensions of edges are 64 for both scalar and vector features. The numbers of heads of both scalar and vector attentions are 4.

We trained Pocket2Mol with batch size 8 and initial learning rate 2×10^{-4} and decayed learning rate by a factor of 0.6 if validation loss did not decrease for 8 validation iterations. We validated the model every 5000 training iterations and the number of total training iterations is $475,000$.

C. The Proof of the equivariant of the vector-attention module

In the vector-attention module (Eq. 9), we generalize the vector inner product of queries and keys to Frobenius inner product. We prove that the vector-attention module still keep the equivariance.

Assume $\vec{\mathbf{z}}_{iq}^{(\mathrm{att})}, \vec{\mathbf{z}}_{ik}^{(\mathrm{att})}$ and $\vec{\mathbf{z}}_{iv}^{(\mathrm{att})}$ are the original queries, keys and values of vector features and A is the vector-attention module: $\vec{\mathbf{z}}_{iq}' = A(\vec{\mathbf{z}}_{iq}^{(\mathrm{att})}, \vec{\mathbf{z}}_{ik}^{(\mathrm{att})}, \vec{\mathbf{z}}_{iv}^{(\mathrm{att})})$. For any rotation or reflection matrix $R \in \mathbb{R}^{3 \times 3}$, following Eq. 9 the attention scores are calculated as:

$$
\begin{aligned}
a_{qk} &= \langle \vec{\mathbf{z}}_{iq}^{(\mathrm{att})} R, \vec{\mathbf{z}}_{ik}^{(\mathrm{att})} R \rangle_F \\
&= \mathrm{tr}\left((\vec{\mathbf{z}}_{iq}^{(\mathrm{att})} R) \cdot (\vec{\mathbf{z}}_{ik}^{(\mathrm{att})} R)^\top \right) \\
&= \mathrm{tr}\left(\vec{\mathbf{z}}_{iq}^{(\mathrm{att})} R R^\top \vec{\mathbf{z}}_{ik}^{(\mathrm{att})\top} \right) \\
&= \mathrm{tr}\left(\vec{\mathbf{z}}_{iq}^{(\mathrm{att})} \vec{\mathbf{z}}_{ik}^{(\mathrm{att})\top} \right) \\
&= \langle \vec{\mathbf{z}}_{iq}^{(\mathrm{att})}, \vec{\mathbf{z}}_{ik}^{(\mathrm{att})} \rangle_F \\
a_{qk}' &= \sigma_{(\mathrm{sf})}(a_{qk})
\end{aligned}
\tag{12}
$$

which are invariant to the transformation R. Then the output vectors of the attention module are:

$$
\vec{\mathbf{z}}_{iq}' = \sum_{v \in \mathcal{V}^{(\mathrm{mol})}} a_{qv}' \vec{\mathbf{z}}_{iv}^{(\mathrm{att})} R = A(\vec{\mathbf{z}}_{iq}^{(\mathrm{att})}, \vec{\mathbf{z}}_{ik}^{(\mathrm{att})}, \vec{\mathbf{z}}_{iv}^{(\mathrm{att})}) R
\tag{13}
$$

which is still equivariant, i.e. $A(\vec{\mathbf{z}}_{iq}^{(\text{att})}R, \vec{\mathbf{z}}_{ik}^{(\text{att})}R, \vec{\mathbf{z}}_{iv}^{(\text{att})}R) = A(\vec{\mathbf{z}}_{iq}^{(\text{att})}, \vec{\mathbf{z}}_{ik}^{(\text{att})}, \vec{\mathbf{z}}_{iv}^{(\text{att})})R$