3D MOLECULAR GENERATION BY VIRTUAL DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Structure-based drug design, i.e., finding molecules with high affinities to the target protein pocket, is one of the most critical tasks in drug discovery. Traditional solutions, like virtual screening, require exhaustively searching on a large molecular database, which are inefficient and cannot get novel molecules beyond the database. The pocket-based 3D molecular generation model, i.e., directly generating a molecule with a 3D structure and binding position in the pocket, is a new promising way to address this issue. However, the method is very challenging due to the complexity brought by the huge continuous 3D space in the pocket cavity. Herein, inspired by Molecular Dynamics, we propose a novel pocket-based 3D molecular generation framework VD-Gen. VD-Gen consists of a Virtual Dynamics mechanism and several carefully designed stages to generate fine-grained 3D molecules with binding positions in the pocket cavity end-to-end. Rather than directly generating or sampling atoms with 3D positions in the pocket like in early attempts, in VD-Gen, we first randomly scatter many virtual particles in the pocket; then with the proposed Virtual Dynamics mechanism, a deep model, acting like a "force field", iteratively moves these virtual particles to positions that are highly possible to contain real atoms. After virtual particles are stabilized in 3D space, we extract the atoms from them. Finally, we further refine the 3D positions of atoms by Virtual Dynamics again, to get a fine-grained 3D molecule. Extensive experiment results on pocket-based molecular generation demonstrate that VD-Gen can generate novel 3D molecules to fill the target pocket cavity with high binding affinities, significantly outperforming previous baselines.

1 INTRODUCTION

Structure-based (pocket-based) drug design, i.e., finding a molecule to fill the cavity of the protein pocket with a high binding affinity [1; 2; 3; 4], is one of the most critical tasks in drug discovery. The most widely used method is virtual screening [5; 6; 7]. Virtual screening iteratively places molecules from a molecular database into the target pocket cavity and evaluates molecules with good binding based on rules such as energy estimation [8; 9; 10; 11]. However, virtual screening is inefficient for the exhaustive search and is infeasible to generate new molecules that are not in the database. Recently, molecular generative models have become a potential solution to address the problem as they could generate novel molecules in an efficient way. The early attempts focused on ligand-based molecular generation[12; 13; 14], which trains models to learn the underlying distribution of the molecules in training data and generate similar molecules. However, those methods didn't consider conditional information, such as the shape of the pocket. Therefore, the generated molecules could hardly fit well with a given pocket in practice. Later, more efforts were paid to studying how to leverage the information of protein pockets for molecular generation. Some pocket-based generative models simply generate molecules in the form of SMILES or graphs [15; 16], without considering the 3D geometric position of the molecule and pocket, which is closely related to binding affinity.

However, directly generating pocket-based molecules in the 3D space is not trivial. Given the 3D structure of a pocket, the ultimate goal of the task is to generate 3D molecules which contain a set of atoms, each with an atom type and the corresponding 3D position. The biggest challenge here is the large space of continuous 3D positions. In most existing generative models (in images/texts), the space of position is usually small and discrete, like an image with 224×224 pixels. To address that, there are some early attempts, which can be roughly categorized into two classes, molecular 3D density grid generation [17] and auto-regressive 3D generation [18; 19; 20]. In 3D density grid generation, similar to images, pockets and molecules are converted to 3D density grids with coarse-grained positions. 3D convolutional models could be used here. But it compresses the

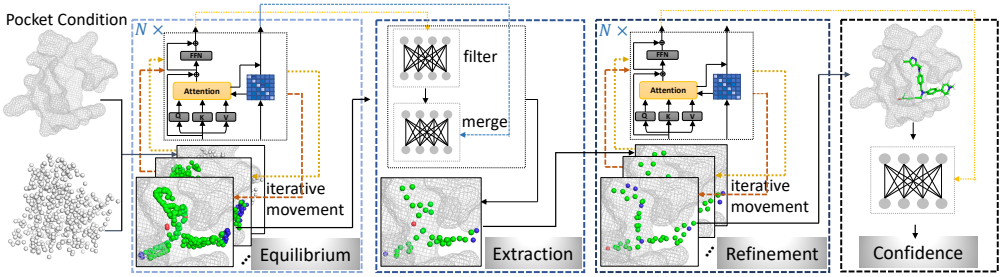


Figure 1: The framework of VD-Gen, which consists of 4 stages, for generating fine-grained 3D molecules with binding positions in the pocket end-to-end. In *Equilibrium* and *Refinement*, the proposed Virtual Dynamics is used to iteratively move the virtual particles.

information of the pocket structure and is hard to generate accurate (fine-grained) molecules due to the coarse-grained grid positions. In auto-regressive 3D generation, an atom (with a 3D position and an atom type) is sampled (or generated) at each time step. But it is very inefficient due to the large sampling space of 3D positions. Besides, using sequential generation for 3D molecules is not reasonable since we do not know which atoms should be generated first.

In short, existing models did not fully tackle the challenges in pocket-based 3D molecular generation. The ideal models should be able to generate fine-grained 3D molecules efficiently, in a one-shot (non-auto-regressive) fashion. To achieve that, we proposed a novel 3D molecular generation framework, VD-Gen, based on a Virtual Dynamics (VD) mechanism. Inspired by the Molecular Dynamics [21; 22], VD contains a deep model, which acts like a "force field", iteratively moving the random scattered "virtual particles" (VPs) to positions that are highly possible to contain real atoms. Based on VD , as illustrated in Fig. 1, VD-Gen framework contains 4 stages to directly generate 3D molecules in the pocket end-to-end. 1) *Equilibrium*. To cover the pocket cavity space as much as possible, many VPs are first randomly placed. Then the VPs are iteratively moved by VD until equilibrium. Ideally, the VPs will be moved into several clusters, each representing a possible atom. 2) *Extraction*. We want to extract atoms from equilibrious VPs in this stage. First, a success rate will be predicted for each VP, a higher success rate means that VP is more close to its target, and the VPs with low success rates will be filtered. Then, a model is used to predict the clustering of VPs, by a pair-wise fashion, and the VPs in the same cluster will be merged into one atom. With the merged atoms, we can get a molecule with a 3D structure. 3) *Refinement*. Although a 3D molecule could be generated in *Extraction* stage, it may be inaccurate due to the error in merging multiple VPs. To get a more accurate 3D molecule, the atoms are iteratively moved by VD again in this step. 4) *Confidence*. A confidence score for the generated 3D molecule will be provided by the model in this stage. The confidence score is instrumental when selecting or ranking from multiple generated results.

Our contributions can be summarized as follows:

- We propose Virtual Dynamics (VD) mechanism, which implicitly simulates Molecular Dynamics by a deep model, iteratively moving the particles to positions that highly possibly contain atoms. We design several strategies, like least-action target assignment and iterative movement, to make the training of VD feasible.
- Although VD can generate rough shapes (densities of particles) of 3D molecules, it is hard to extract molecules from the shapes. To address the problem, we further propose a novel pocket-based 3D molecular generation framework VD-Gen, which end-to-end extracts a 3D molecule from many particles, then refines it by VD again, and predicts a confidence score used for selecting or ranking.
- Under VD-Gen, to tackle the limited data in pocket-based 3D molecular generation, we design a self-partial-generation pretraining task and successfully use it to further improve performance.
- Multiple evaluation metrics, such as Vina [23], MM-PBSA [24], 3D Similarity [25], are used to benchmark VD-Gen thoroughly. Experiments results demonstrate that VD-Gen can generate diverse drug-like molecules with high binding affinities, significantly outperforming all baselines. Ablation studies and case studies are designed to further demonstrate the effectiveness of VD-Gen.

2 METHOD

The problem of pocket-based 3D molecular generation could be denoted as $\mathbf{M} = h(\mathbf{P}; \boldsymbol{\theta})$, where $h(\cdot; \boldsymbol{\theta})$ is the model with learnable parameter $\boldsymbol{\theta}$, $\mathbf{P} = \{(\boldsymbol{x}_i^p, \boldsymbol{y}_i^p)\}_{i=1}^u$ is the set of u atoms in the pocket, $\boldsymbol{x}_i^p \in \mathbb{R}^t$ and $\boldsymbol{y}_i^p \in \mathbb{R}^3$ are the i-th pocket atom's type (one-hot) and coordinate, respectively, t is the number of atom types, and $\mathbf{M} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^m$ is the set of m atoms of the generated molecule.

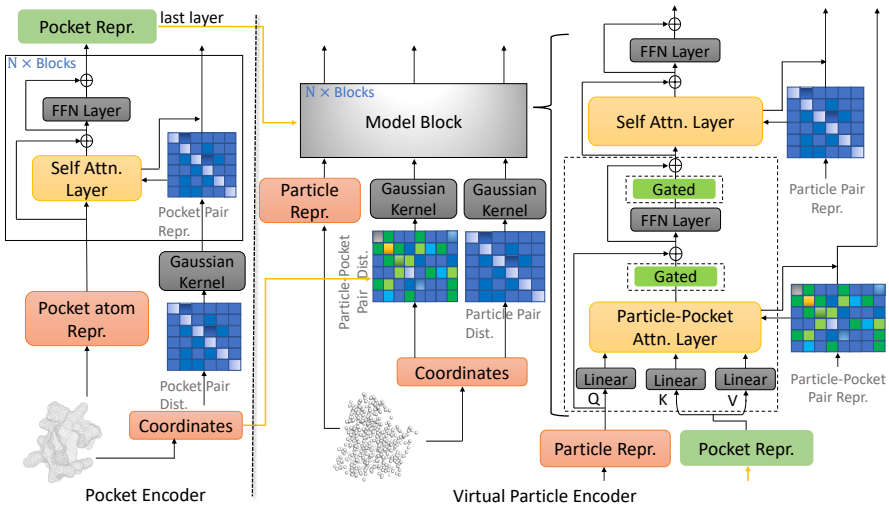


Figure 2: The backbone model used in VD-Gen. Details are in Appendix A.1.1 and Alg. 2.

2.1 Virtual Dynamics

As aforementioned, directly generating $\mathbf{M}$ is challenging due to the large space of 3D positions. Therefore, inspired by Molecular dynamics (MD), we propose Virtual Dynamics (VD), which iteratively refines the particles from a random state, rather than direct generation. MD is a Newtonian Mechanics based computational simulation to move atoms or other microscopic particles. In MD, what determines how atoms move is the molecular force field, a physical model that defines the interactions between atoms. The potential energy surface (PES) [26; 27], a function of energy based on atomic positions, is used to describe the energy landscape of the system. Each minimal energy on PES corresponds to a physical stable state, in which the atoms prefer to stay in particular positions.

Virtual Dynamics (VD) contains a deep model, acting like a "force field", implicitly predicting the preferred positions of ligand molecular atoms in the pocket cavity by moving the "virtual particles"(VPs) toward those positions. Formally, VD could be denoted as $\mathbf{V}_r = h(\mathbf{V}_0, \mathbf{P}, r; \boldsymbol{\theta})$, where r is the number of rounds, $\mathbf{V}_r = \{(\boldsymbol{x}_i^r, \boldsymbol{y}_i^r)\}_{i=1}^n$ is the set of n VPs that are generated at the r-th round. Here we define the VPs as particles without fixed atom types. Besides, VD can generate more VPs than the real atoms, i.e., $n = |\mathbf{V}_r^n|$ can be larger than $m = |M|$, and VPs can overlap with each other. To train a model to achieve effective movement of VPs, VD consists of 4 parts: 1) *Backbone Model*, a SE(3) model takes $\mathbf{V}_r$ as input and outputs the refined $\mathbf{V}_{r+1}$; 2) *Target Assignment*, a method to assign targets for VPs during training; 3) *Iterative Movement*, a strategy to update positions of VPs iteratively like MD; 4) *Training Objectives*, effective objective functions to train VD .

Backbone Model We can denote the model as $\mathbf{V}_{r+1} = f(\mathbf{V}_r, \mathbf{P}; \boldsymbol{\theta})$. To predict the coordinates effectively, the model f should be SE(3)-equivariance. We mainly follow the design of the efficient SE(3)-equivariance Transformer proposed in Uni-Mol [28] and Graphormer-3D [29]. However, they did not consider the interaction between pocket and molecule. Therefore, as illustrated in Fig. 2, we extend the model by adding an additional pocket encoder, and a particle-pocket attention layer to capture the interactions between pocket atoms and VPs. In particular, the key/value in the particle-pocket attention is from the node representation of the *last* layer in the pocket encoder. Besides, to encode the 3D spatial interactions between the pocket and VPs, the pair distance between pocket atoms and VPs is used for particle-pocket spatial position encoding. For efficiency purposes, particle-pocket attention is only used in every 4-layer, not in all layers.

To encode 3D positions, we follow Uni-Mol and use SE(3)-invariant Gaussian kernel to encode the pair-wise Euclidean distances, as shown in Fig. 2. To predict 3D positions directly, the SE(3)-equivariance coordinate head in Uni-Mol [28] is used. Besides, to predict the atom types of particles after movement, an atom type prediction head is introduced. Due to space restrictions, we leave the details of the above components in Appendix A.1.1.

Target Assignment The goal of model f is to move the VPs to the preferred positions of ligand molecular atoms. To achieve this, we can directly assign a real atom as the training target for each VP. Formally, given the ground-truth atoms $\mathbf{G} = \{(\boldsymbol{x}_i^g, \boldsymbol{y}_i^g)\}_{i=1}^m$ and the random initialized VPs $\mathbf{V}_0$,

Algorithm 1 Iterative Movement

Require: R: max rounds, $\mathbf{P}$: pocket atoms, $\mathbf{V}_0$: random initialized virtual particles

$r \leftarrow$ uniform(1, R) if *training* else R ▷ Sampling is only enabled at training
disable_gradient() ▷ Disable gradient calculation globally
for $k \in [1, ..., r-1)$ **do** ▷ Iterative updates without gradients
  $\mathbf{V}_k \leftarrow f(\mathbf{V}_{k-1}, \mathbf{P}; \boldsymbol{\theta})$ ▷ backbone model f predict the types and positions
enable_gradient() ▷ Enable gradient calculation globally
$\mathbf{V}_r \leftarrow f(\mathbf{V}_{r-1}, \mathbf{P}; \boldsymbol{\theta})$ ▷ update with gradients
return $\mathbf{V}_r$ ▷ Return the positions and types of particles

there are n^m possible assignments. Following the principle of least action [30], the assignment with minimal moving distance is favored. That is to optimize Min $\sum_{i=1}^{n} \|\boldsymbol{y}_i^0 - \boldsymbol{y}_{a_i}^g\|_2$, where $\boldsymbol{x}_i^0$ is the initial position and $a_i \in \mathbb{N}$ is the assigned target for i-th VP. This optimization problem is easy to solve: for each VP, assign its nearest real atom as the training target, i.e., $a_i = \arg\min_{j=1}^{m} \|\boldsymbol{y}_i^0 - \boldsymbol{y}_j^g\|_2$. However, by this method, some real atoms may not be assigned as targets when VPs are closer to other real atoms. To increase the coverage, there are two methods. The first is taking the coverage as a constraint in the above optimization problem; the second is using more VPs to cover the pocket cavity as possible. Although the former is more favorable from the algorithm perspective, it increases the learning difficulty of VP movement since the constrained assignment breaks the principle of least action [1]. Therefore, in VD , we take the latter one, using more VPs to ensure coverage. Besides, we propose an algorithm to determine the pocket cavity, and VPs are only initialized inside the detected cavity. In particular, we use a breadth-first search algorithm, from a given position in the cavity, to detect the cavity space. Details are in Appendix A.1.2.

Iterative Movement Since VPs are scattered randomly in the pocket cavity, there is a wide range of distances between each VP and its target. As a result, it is not realistic that every VP can be moved to the right target position in one step. Therefore, VD takes a strategy that iteratively moves the VPs and predicts their types with multiple rounds. In particular, at each round, the model will take the VPs' positions and types from the previous round as inputs, and output the new positions and types for them. In the first few rounds, the VPs that are close to the target positions will soonly approach their target positions. But for the VPs that are far away from the target positions, it may take more rounds to approach. In short, with iterative movement, more VPs could reach their target positions.

However, training the model with multiple rounds is not efficient in both speed and memory consumption. To reduce the training cost, we adopt the stochastic iteration in AlphaFold2 [31]. In particular, during training, the number of rounds r is uniformly sampled between 1 and R, where R is the max round. Then, the model is run on the forward-only mode in the first $r-1$ rounds, without loss calculation and gradient backward. Finally, the gradient and backward are enabled at the r-th round. During inference, the sampling on rounds is not used. The above algorithm is shown in Alg. 1.

In addition, we propose two technologies to improve training stability. First, the SE(3) coordinate head is initialized to predict the zero delta positions; thus, the predicted movements of VPs are nearly zeros at the beginning of training, and gradually increase. Second, to avoid moving too fast in each round, a regularization of the delta distance between the input and output positions is used during training.

Training Objectives With assigned targets ($a_i = \arg\min_{j=1}^{m} \|\boldsymbol{y}_i^0 - \boldsymbol{y}_j^g\|_2$), the training of VD is straightforward. First, a clip L2 loss is used for the coordinate prediction, clipping is for the training stability. Second, as aforementioned, a regularization of the moving distance of iterative movement is introduced to avoid moving too fast and improve training stability. Third, a negative log likelihood loss is used for the particle type prediction. Finally, two auxiliary L1 losses are used for the particle-particle pair distance prediction and particle-pocket pair distance prediction, respectively. The final training objective loss function could be denoted as:

$$\mathcal{L}_{\mathcal{VD}} = \frac{1}{n}\sum_{i=1}^{n}\Big(\text{clip}(||\boldsymbol{y}_i^r - \boldsymbol{y}_{a_i}^g||_2, \tau) + \max(||\boldsymbol{y}_i^r - \boldsymbol{y}_i^{r-1}||_2 - \delta, 0) + \text{NLL}(\bar{\boldsymbol{x}}_i^r, \boldsymbol{x}_{a_i}^g) + \frac{1}{n}\sum_{j=1}^{n}||\boldsymbol{d}_{ij}^r - \boldsymbol{d}_{a_i,a_j}^g||_1 + \frac{1}{u}\sum_{j=1}^{u}||\boldsymbol{c}_{ij}^r - \boldsymbol{c}_{a_i,j}^g||_1\Big), \tag{1}$$

[1] Our early attempts on target assignment are described in Appendix B.4.

where r is the number of rounds, $\bar{\boldsymbol{x}}_i^r$ is the predicted atom type distribution of i-th particle, $\boldsymbol{d}_{ij}^r$ ($\boldsymbol{d}_{a_i,a_j}^g$) is the predicted (ground-truth) distance of the i-th and j-th particle pair, $\boldsymbol{c}_{ij}^r$ ($\boldsymbol{c}_{a_i,j}^g$) is the predicted (ground-truth) distance of the i-th particle and the j-th pocket atom.

2.2 VD-GEN FRAMEWORK

With VD , a rough shape (density of VPs) of the 3D molecule could be formed by the VPs after iterative movement. We may use some rule-based solutions, like clustering by distances, to extract 3D molecules from the VPs. However, rule-based solutions are not end-to-end and could fail in various scenarios. Therefore, to better leverage VD , we further develop an end-to-end pocket-based 3D molecular generation framework, called VD-Gen, with the following 4 stages, illustrated in Fig 1.

Equilibrium This stage is exactly the same as the VD . Many VPs are first uniformly scattered in the pocket. Then, VPs iteratively move toward their target positions. Finally, VPs will reach a stable state.

Extraction With the equilibrious VPs, a rough shape of the 3D molecule could be got, and we want to extract a 3D molecule from it. For the end-to-end purpose, we propose a deep model based solution to extract atoms. Formally, the model can be denoted as $\mathbf{W}_0 = h_{ex}(\mathbf{V}_r, \mathbf{P}; \boldsymbol{\theta}_{ex})$, where $\boldsymbol{\theta}_{ex}$ is learnable parameters, $\mathbf{W}_0 = \{(\hat{\boldsymbol{x}}_i^0, \hat{\boldsymbol{y}}_i^0)\}_{i=1}^m$ is the set of m VPs after extraction. The model reduces VPs from n to m by two steps, filter and merge. First, as some VPs may fail to approach their target positions, we want to filter out them. A binary classification head is used to predict the success rates of VPs, and the training targets are "success" if the distances between VPs and their target positions after *Equilibrium* are smaller than a threshold. And we filter out the VPs based on the predicted success rate.

Second, we want to merge the remaining VPs into atoms. Based on the pair representation of VPs, we use another binary classification head to predict whether a VP pair should be merged or not. Ideally, the VPs with the same target atom should be merged, thus training label for a VP pair with the same atom target is set to "true". When there are n VPs and m real atoms, the ratio of "true" class is about $\frac{m\times(n/m)^2}{n^2} = \frac{1}{m}$. As m ranges from dozens to hundreds, the binary classification task here is very unbalanced. Thus, we introduce a focal loss [32] to balance the classes. With the predicted pair-wise merge probability matrix, we can use a threshold to get a binary merge matrix and merge VPs into clusters according to the matrix. However, it is hard to decide a threshold since the training of pair-wise merge is unbalanced. To tackle that, we further introduce a prediction task for the number of ligand molecular atoms, based on pocket atom representation. During inference, we use binary search to find a merging threshold that satisfies the predicted atom number, details are in Appendix A. There will be several (ideally m) merge clusters, and we denote $\boldsymbol{w}_i$ as the set of the indices of i-th cluster's VPs. Then, to initialize $\mathbf{W}_0$, we use $\hat{\boldsymbol{x}}_i^0 = \text{Uniform}(\{\boldsymbol{x}_j^r | j \in \boldsymbol{w}_i\})$ and $\hat{\boldsymbol{y}}_i^0 = \text{Mean}(\{\boldsymbol{y}_j^r | j \in \boldsymbol{w}_i\})$, to sample an atom type and get an average coordinate respectively.

Refinement After *Extraction*, a 3D molecule with a set of VPs ($\mathbf{W}_0$) could be formed. However, due to the possible error in *Extraction*, the predicted 3D molecule may not be very accurate. To get a more accurate 3D molecule ($\mathbf{W}_r$), we use VD again, with different model weights, to iteratively move these VPs to their target positions. Different with *Equilibrium*, the training target assignment of the i-th VP is the most frequent target atom in the cluster $\boldsymbol{w}_i$, not its nearest atom.

Confidence Abundant molecules are usually generated in real-world tasks. We want to select or rank the molecules according to binding affinities. Although we can use computational simulations or wet experiments to examine the generated molecules, they are too costly, especially for a large number of molecules. To further improve the usability of VD-Gen and reduce the extra cost of selecting good molecules, we explicitly train a task to learn the confidence scores for the generated molecules. In particular, following AlphaFold [31], we compute the LDDT score [33] of the generated molecule and ground-truth molecule, and a pLDDT(predict LDDT) head is used to learn the LDDT score. During inference, the output of pLDDT head is used as the confidence score of the generated molecule.

The loss functions in the above 4 stages are combined, and the entire VD-Gen framework is trained end-to-end. Due to space restrictions, we leave the details of the above loss functions in Appendix A.

2.3 PRE-TRAINING FOR VD-GEN

Due to the limited protein-ligand binding data for the supervised training, VD-Gen may fail to train or overfit the small training data. Therefore, to improve the model ability, we pretrain the pocket encoder and the VP encoder by large-scale unlabeled data, respectively. The pretrained pocket encoder is directly taken from the pretrained one from Uni-Mol [28]. For the VP encoder, the pretraining is mostly the same as the VD-Gen framework, except that pocket is not involved. In particular, the

pocket related components, like particle-pocket attention, are all removed. Nevertheless, without pocket as a condition, the training of VD-Gen is infeasible. So we design a self-partial-generation pretraining task, by using a part of the molecule as the known condition and generating the unknown part. To be more consistent with finetuning, only a few atoms, about 20% to 30%, are kept as a condition. To have a continuous 3D space for VPs to generate, we randomly remove the atoms in the continuous region. We use a greedy recursive algorithm to find a cluster of atoms to remove, and then, the VPs are randomly scattered in the continuous region of these removed atoms.

During finetuning, the backbone model in VD-Gen loads the weights from two pretrained models. For the VP encoder, the weights in the particle-pocket attention layers are not pretrained and are randomly initialized. Gated layers, initialized as zeros, are used in the residual connections of particle-pocket attention layers. Therefore, the outputs of random initialized particle-pocket attention layers will not affect the pretrained encoders at the beginning of finetuning.

2.4 Extending VD-Gen to Pocket-based 3D Molecular Optimization

Molecular optimization is also an important task in real-world drug design. In molecular optimization, rather than generating from scratch, the goal is to replace a part of the given molecule, like a fragment, and to get a molecule with better binding affinity. Here, we extend VD-Gen to the pocket-based 3D molecular optimization. In particular, as illustrated in Fig. 8, we first randomly remove a fragment of the given molecule, and the model is learned to generate it, with the pocket and the remaining atoms in the molecule as conditions. In this way, although it is not trained to optimize molecules directly, the model learns how to remove-then-fill a fragment of a molecule, and thus could be used in molecular optimization tasks. The benchmark results of molecular optimization are left to Appendix B.8.

3 Experiments

3.1 Settings

Evaluation metrics There is not a golden metric to evaluate the generated molecules, so we use multiple metrics to have a comprehensive evaluation. 1) *3D Similarity*. As the pocket-based 3D generation models are trained by the 3D structures of the pockets and molecules, the most direct metric to examine the models' generative ability is to evaluate the 3D similarity between the generated molecule and the ground-truth one. Here we use LIGSIFT [25] to calculate the overlapping ratio in 3D space between two molecules. 2) *Vina*. Docking scores, like Vina [23], are widely used in previous pocket-based generation works, for they are easy to compute. To be consistent with previous works, we also use **Vina** as a metric. However, previous works usually relied on Vina's re-docking, in which the molecular conformation and binding pose may be largely changed by docking tools. Thus, to directly evaluate the 3D molecules generated by model, we add an additional **Vina*** score that does not use re-docking. 3) *MM-PBSA*. Although docking scores are easy and fast to compute, they are proposed to recall the possible hits in the large-scale virtual screening, not for ranking. Thus, docking scores are not good metrics to compare the binding affinities for different models [34], and we further use the slower but more accurate MM-PBSA (Molecular Mechanics Poisson–Boltzmann Surface Area) [35] as a metric. Based on MM-PBSA, we add two additional metrics. **MM-PBSA B.T.** (MM-PBSA Better than Target), which computes the percentage of generated molecules with better MM-PBSA scores than ground-truth. **MM-PBSA Rank**, which computes the average rankings of different models among different complexes. Due to MM-PBSA scores varying largely in different complexes, MM-PBSA Rank can better compare different models. The details of the above metrics are described in Appendix B.2.

Data We use 3D molecular conformations from Uni-Mol [28] to pretrain VP encoder. PDBBind 2020 dataset [36; 37], containing 19,443 protein-ligand complexes crystal structures, are used to finetune the VD-Gen. Although the cross-docked data [38] used in the previous works is much larger, it is built by docking tools and thus is not accurate as PDBBind, so we do not use it. For the test set, we use 100 protein-ligand complex crystal structures from [24], on which MM-PBSA was validated to be effective. To avoid leakage, we remove the training data's complexes whose protein sequences are similar to the ones in the test set. In particular, two protein sequences are identified as similar if their e-value from BLAST [39] search results is larger than 0.4. There are 18,413 training complexes after filtering.

Training We leave the detailed hyper-parameters used in training to Appendix B.1.

3.2 Molecule Generation Performance

Baselines We compare VD-Gen with several previous 3D pocket-base molecular generation models: the 3D density generation model LiGAN [17], and the auto-regressive 3D generation models

Table 1: Performance on pocket-based 3D molecular generation.

Model	3D Sim(↑)	Vina(↓)	Vina*(↓)	MM-PBSA(↓)	MM-PBSA-Rank(↓)	MM-PBSA-B.T.(%↑)
LiGAN[17]	0.356	-6.724	-5.372	-17.865	2.57	0.3
3DSBDD[18]	0.365	-8.662	-7.227	-30.221	2.26	3.2
GraphBP[19]	0.333	-8.710	-3.689	-5.130	3.98	0
Pocket2Mol[20]	0.352	-8.332	-6.525	-7.823	3.49	0
VD-Gen	**0.414**	**-9.047**	**-7.444**	**-51.258**	**1.18**	**13.5**

Table 2: Ablation study on pretraining.

Setting	no pretrain	pretrain pocket encoder only	pretrain VP encoder only	pretrain
3D Similarity (↑)	0.361	0.379	0.402	0.414

GraphBP [19], 3DSBDD [18], and Pocket2Mol [20]. For all models, we generate 500 results for each pocket, and then select 100 from them. For 3DSBDD and Pocket2Mol, beam search is used and the top 100 results are selected. For VD-Gen, the selection is based on the confidence score. For LiGAN and GraphBP, random 100 results are selected due to they did not implement beam search.

Results As we pay more attention to the generated molecules with high binding affinities, we report the top 5-th percentile result for Vina, Vina*, and MM-PBSA. MM-PBSA-Rank is calculated based on the top 5-th percentile MM-PBSA result. The 10-th, 25-th, and 50-th percentile results are in Appendix B.3.

From the results in Table 1, it is easy to conclude: 1) VD-Gen significantly outperforms all other baselines in all metrics, with top-1 MM-PBSA Rank, demonstrating the superior performance of the proposed VD-Gen. 2) MM-PBSA B.T shows that VD-Gen can generate more molecules with better MM-PBSA scores than the ground-truth ones, while baseline hardly can. 3) In 3D Similarity results, VD-Gen also largely outperforms baselines, indicating that VD-Gen effectively learned the pocket-based 3D molecular generation and can generalize to unseen pockets. 4) Although some baselines achieve good performance on Vina scores, like GraphBP and Pocket2Mol, their Vina* and MM-PBSA scores are very poor. We believe the re-docking in Vina fixes their generated 3D structures and then a good Vina score could be obtained. This result indicates that the previously widely used Vina score is not a good metric for pocket-based 3D molecular generation.

3.3 Ablation Study

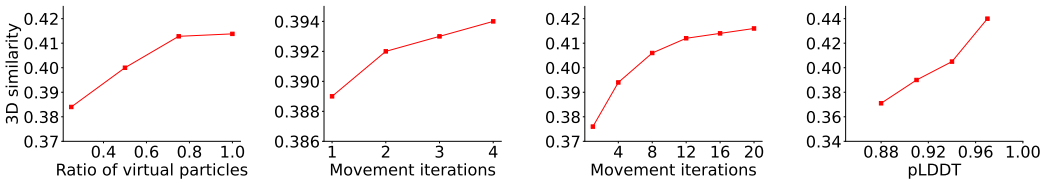
Pocket coverage As discussed in Sec. 2.1, Virtual Dynamics requires many VPs to cover the pocket cavity as much as possible. And we study how the number of VPs affects the final performance, the results are shown in Fig. 3(a). From the result, it is clear that the number of VPs will affect the performance, and the results with more VPs are better.

Effectiveness of Refinement stage The *Refinement* stage is used to further refine the 3D molecule after *Extraction*. To examine how *Refinement* affects the final performance, we benchmarked different movement iterations in *Refinement*. As shown in Fig. 3(c), we can find the results with more iterations are better. The result indicates the necessity of the *Refinement* stage.

Iterative Movement ronuds Iterative Movement is critical in the Virtual Dynamics. From the result in Fig. 3(c), we can find the results with more rounds are better. We also benchmark the effectiveness of Iterative Movement in *Equilibrium* stage. And we reduce the movement iterations to 25% in *Refinement* stage, to better show the impact brought by *Equilibrium* stage. As shown in Fig. 3(b), we can find more iteration rounds in *Equilibrium* also improves the final performance.

Effectiveness of Confidence stage The pLDDT score is outputted at *Confidence* stage, and used for selecting or ranking molecules, and we want to check its effectiveness. In particular, we calculate the correlation between 3D similarity and the pLDDT for the generated molecules on a pocket (PDBID 1I7Z), and the result is shown in Fig. 3(d). It is clear that with a larger pLDDT score, the corresponding 3D Similarity is better. This result indicates that the confidence score provided by VD-Gen is effective to select or rank the generated molecules.

Effectiveness of pretraining We also benchmark the performance brought by pretraining. In particular, we add three additional models, one without any pretraining, one only with pocket



(a) Pocket Coverage (b) Equilibrium (c) Refinement (d) Confidence

Figure 3: Ablation studies for VD−Gen.

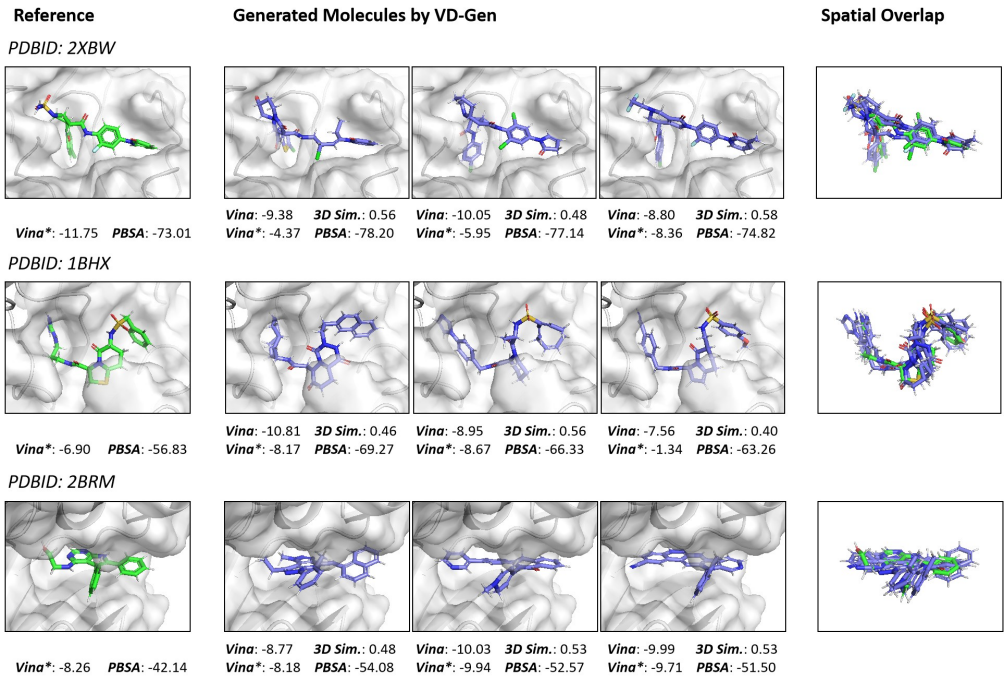


Figure 4: Generated molecules with high 3D similarity to the reference molecular and high PBSA scores for three protein pockets. Gray surfaces are the protein pockets. Green molecules are the ground truth molecules. Purple molecules are the molecules generated by VD−Gen. Lower Vina score, lower PBSA score and higher 3D similarity indicate higher binding affinity.

pretraining, and one only with particle pretraining. From the results shown in Table 2, we can easily conclude that pretraining indeed boosts the performance of VD−Gen.

3.4 Case Study

Here, we selected three protein pockets from the test set to visualize the generated results of VD−Gen on pocket-based generation tasks. As shown in Fig 4, for each pocket, 3 molecules (purple molecules in the middle column) with the top MM-PBSA scores are selected for display. These molecules are shown as they as, without any structural post-processing. Green molecules are the ground truth molecules, and the rightmost column is the spatial overlapping of the generated molecules and the original molecule.

In the first case (PDBID: 2XBW), the protein pocket has a pit deep inside the protein (bottom left of the image), the volume of which can accommodate about one benzene ring. It is a challenging task due to the small size of the pit and the long distance from the center of the whole pocket. We can see that the molecules generated by VD−Gen have successfully grown fragments within the pit. On the other hand, the three generated molecules have good 3D similarity with the original molecules, and the MM-PBSA score is good, the Vina scores of the original molecule are much better than those of the three generated molecules. If we only use Vina to pick molecules, It may lead to not picking good molecules.

In the second case (PDBID: 1BHX), the protein pocket is bulky, which requires the generation of protein-interacting fragments at both ends of the protein pocket, and connecting the two ends

together by a molecular backbone, we can see the original molecule is long and distorted, making it a challenging prediction task. We see that the molecules generated by VD-Gen replicate the shape of the original molecules well, filling the uneven protein pockets well. All three molecules have good 3D similarity and MM-PBSA scores.

In the third case (PDBID: 2BRM), the protein pocket is flat, which requires that the molecular backbone of the ligand bound to it should be close to a planar structure, such as composed of conjugated aromatic rings. We can see that the molecules generated by VD-Gen are the same as the original molecular structures, whose molecular backbone is a planar structure composed of conjugated aromatic rings, and the part toward the outside of the pocket is flexible. We can see that in this case, Vina scoring, MM-PBSA scoring, and 3D similarity all show good agreements.

From these three cases in Fig 4, we can see that VD-Gen has demonstrated good generation capabilities on different types of challenging molecular generation tasks. For example, the generated molecules can fill deep pockets, follow the trend of large pockets, or match the special structure of the pockets, and the 3D similarity between the generated molecule and the molecule in the original crystal structure is high. On the other hand, we can see that the MM-PBSA score and 3D similarity maintain good consistency in evaluating the quality of generated molecules, while the Vina score fails in some cases, which indicates that it is unreasonable to select molecules based on the Vina score alone.

4 Related Work

Ligand-Based Molecular Generation Early works focused on ligand-based molecular generation, took a set of molecules as training data, and generated molecules based on the learned distribution of training data. And these methods mainly represented molecules as 1D SMILES strings and 2D molecular graphs, and used VAEs [12; 13; 14; 40; 41; 42], GANs [43; 44], flow models [45] for one-shot generation, RNNs [46; 47; 48; 49], reinforcement learning approaches[50; 51] for step-by-step generation. And some works [52; 53; 54] tried to preserve structural features like molecular scaffolds, or physicochemical properties like QED, to gain better generated molecules compared to randomly generation. However, those methods did not take the binding affinity against a specific protein pocket as a target directly thus the generated molecules hardly worked well in real-world tasks. Some recent works [55; 56; 57; 58] also tried the ligand-based 3D molecular generation.

Pocket-Based Molecular Generation Due to the importance of binding affinity in drug design, recent works involved the information of protein pockets for molecular generation. Early attempts [15; 16] encoded pocket information and took it as a condition to generate molecules in SMILES strings or molecular graphs. However, since the binding affinity depends on the spatial positions of pocket and molecule, the latter works paid more effort in generating molecules with 3D spatial structures. Some works [17], recognized as molecular 3D density grid generation, converted pockets and molecules into 3D density grids, and applied 3D convolutional models like processing images. But as the pocket cavity is large, the positions of pockets and molecules are coarse-grained in 3D density grids and it leads to information loss and hard to generate fine-grained molecules. Besides, it is not end-to-end since the conversion from 3D density to 3D coordinates is required and usually causes additional accuracy loss. Some other works [18; 19; 20], recognized as auto-regressive 3D molecular generation, sampled/generated atoms in 3D space one by one to form a molecule. Suffering from the large space of continuous 3D positions, it is quite inefficient. Besides, unlike the sequential nature in text, the atoms in a molecule do not have a sequential order. That is, we do not know which atoms should be generated first, and thus, using auto-regressive generation for 3D molecules is not reasonable.

5 Conclusion

In this paper, we propose VD-Gen, a novel pocket-based 3D molecular generation framework, which consists of a Virtual Dynamics mechanism and several stages, to generate fine-grained 3D molecules with good binding affinities against the pocket end-to-end. In particular, with Virtual Dynamics, many virtual particles are first randomly scattered in the pocket cavity, and are iteratively moved to positions that are highly possible to contain real atoms. Then, a coarse-grained 3D molecule could be extracted by deep models from these particles. Next, the 3D molecule is continued refined by Virtual Dynamics again, and a fine-grained 3D molecule could be obtained. Finally, a confidence score will be calculated for the generated molecule for the need of selecting or ranking. Several strategies are proposed to make the training of VD-Gen feasible. Experiment results demonstrate that VD-Gen can generate molecules with higher binding affinities to protein pockets and more accurate 3D binding structures than other baselines. Several case studies also demonstrate the effectiveness of VD-Gen.

REFERENCES

[1] Hugo Kubinyi. *3D QSAR in drug design: volume 1: theory methods and applications*, volume 1. Springer Science & Business Media, 1993.

[2] Renee L DesJarlais, Robert P Sheridan, George L Seibel, J Scott Dixon, Irwin D Kuntz, and R Venkataraghavan. Using shape complementarity as an initial screen in designing ligands for a receptor binding site of known three-dimensional structure. *Journal of medicinal chemistry*, 31(4):722–729, 1988.

[3] Robert S DeWitte, Alexey V Ishchenko, and Eugene I Shakhnovich. Smog: de novo design method based on simple, fast, and accurate free energy estimates. 2. case studies in molecular design. *Journal of the American Chemical Society*, 119(20):4608–4617, 1997.

[4] Robert S DeWitte and Eugene I Shakhnovich. Smog: de novo design method based on simple, fast, and accurate free energy estimates. 1. methodology and supporting evidence. *Journal of the American Chemical Society*, 118(47):11733–11744, 1996.

[5] W Patrick Walters, Matthew T Stahl, and Mark A Murcko. Virtual screening—an overview. *Drug discovery today*, 3(4):160–178, 1998.

[6] Brian K Shoichet. Screening in a spirit haunted world. *Drug discovery today*, 11(13-14):607–615, 2006.

[7] Brian K Shoichet. Virtual screening of chemical libraries. *Nature*, 432(7019):862–865, 2004.

[8] Anita de Ruiter and Chris Oostenbrink. Free energy calculations of protein–ligand interactions. *Current opinion in chemical biology*, 15(4):547–552, 2011.

[9] Christophe Chipot and Andrew Pohorille. *Free energy calculations*, volume 86. Springer, 2007.

[10] Clara D Christ, Alan E Mark, and Wilfred F Van Gunsteren. Basic ingredients of free energy calculations: a review. *Journal of computational chemistry*, 31(8):1569–1582, 2010.

[11] Julien Michel and Jonathan W Essex. Prediction of protein–ligand binding affinity by free energy simulations: assumptions, pitfalls and expectations. *Journal of computer-aided molecular design*, 24(8):639–658, 2010.

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

[13] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*, 2018.

[14] Robin Winter, Floriane Montanari, Andreas Steffen, Hans Briem, Frank Noé, and Djork-Arné Clevert. Efficient multi-objective molecular optimization in a continuous latent space. *Chemical science*, 10(34):8016–8024, 2019.

[15] Miha Skalic, Davide Sabbadin, Boris Sattarov, Simone Sciabola, and Gianni De Fabritiis. From target to drug: generative modeling for the multimodal structure-based ligand design. *Molecular pharmaceutics*, 16(10):4282–4291, 2019.

[16] Mingyuan Xu, Ting Ran, and Hongming Chen. 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.

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

[18] Shitong Luo, Jiaqi Guan, Jianzhu Ma, and Jian Peng. A 3d molecule generative model for structure-based drug design. *arXiv preprint arXiv:2203.10446*, 2022.

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

[20] Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. *arXiv preprint arXiv:2205.07249*, 2022.

[21] Berni J Alder and Thomas Everett Wainwright. Studies in molecular dynamics. i. general method. *The Journal of Chemical Physics*, 31(2):459–466, 1959.

[22] Berni Julian Alder and Thomas Everett Wainwright. Studies in molecular dynamics. ii. behavior of a small number of elastic spheres. *The Journal of Chemical Physics*, 33(5):1439–1451, 1960.

[23] 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.

[24] Maohua Yang, Dongdong Wang, and Hang Zheng. Uni-gbsa: An automatic workflow to perform mm/gb(pb)sa calculations for virtual screening. *ChemRxiv*, 2022.

[25] Ambrish Roy and Jeffrey Skolnick. Ligsift: an open-source tool for ligand structural alignment and virtual screening. *Bioinformatics*, 31(4):539–544, 2015.

[26] Alan D McNaught, Andrew Wilkinson, et al. *Compendium of chemical terminology*, volume 1669. Blackwell Science Oxford, 1997.

[27] Shin Sato. On a new method of drawing the potential energy surface. *The Journal of chemical physics*, 23(3):592–593, 1955.

[28] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. 2022.

[29] Yu Shi, Shuxin Zheng, Guolin Ke, Yifei Shen, Jiacheng You, Jiyan He, Shengjie Luo, Chang Liu, Di He, and Tie-Yan Liu. Benchmarking graphormer on large-scale molecular modeling datasets. *arXiv preprint arXiv:2203.04810*, 2022.

[30] Richard Feynman. *The Character of Physical Law, with new foreword*. MIT press, 2017.

[31] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[33] Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. lddt: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, 29(21):2722–2728, 2013.

[34] Tiejun Cheng, Xun Li, Yan Li, Zhihai Liu, and Renxiao Wang. Comparative assessment of scoring functions on a diverse test set. *Journal of chemical information and modeling*, 49(4):1079–1093, 2009.

[35] Samuel Genheden and Ulf Ryde. The mm/pbsa and mm/gbsa methods to estimate ligand-binding affinities. *Expert opinion on drug discovery*, 10(5):449–461, 2015.

[36] Renxiao Wang, Xueliang Fang, Yipin Lu, and Shaomeng Wang. The pdbbind database: Collection of binding affinities for protein- ligand complexes with known three-dimensional structures. *Journal of medicinal chemistry*, 47(12):2977–2980, 2004.

[37] Renxiao Wang, Xueliang Fang, Yipin Lu, Chao-Yie Yang, and Shaomeng Wang. The pdbbind database: methodologies and updates. *Journal of medicinal chemistry*, 48(12):4111–4119, 2005.

[38] 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.

[39] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L Madden. Blast+: architecture and applications. *BMC bioinformatics*, 10(1):1–9, 2009.

[40] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2):577–586, 2020.

[41] Orion Dollar, Nisarg Joshi, David AC Beck, and Jim Pfaendtner. Attention-based generative models for de novo molecular design. *Chemical Science*, 12(24):8362–8372, 2021.

[42] André F Oliveira, Juarez LF Da Silva, and Marcos G Quiles. Molecular property prediction and molecular design using a supervised grammar variational autoencoder. *Journal of Chemical Information and Modeling*, 62(4):817–828, 2022.

[43] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

[44] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L Guimaraes, and Alan Aspuru-Guzik. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). 2017.

[45] 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.

[46] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

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

[48] 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.

[49] Daniel Flam-Shepherd, Kevin Zhu, and Alán Aspuru-Guzik. Keeping it simple: Language models can learn complex molecular distributions. *arXiv preprint arXiv:2112.03041*, 2021.

[50] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

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

[52] Yibo Li, Jianxing Hu, Yanxing Wang, Jielong Zhou, Liangren Zhang, and Zhenming Liu. Deepscaffold: a comprehensive tool for scaffold-based de novo drug discovery using deep learning. *Journal of chemical information and modeling*, 60(1):77–91, 2019.

[53] Jaechang Lim, Sang-Yeon Hwang, Seokhyun Moon, Seungsu Kim, and Woo Youn Kim. Scaffold-based molecular design with a graph generative model. *Chemical science*, 11(4):1153–1164, 2020.

[54] 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.

[55] Vitali Nesterov, Mario Wieser, and Volker Roth. 3dmolnet: a generative network for molecular structures. *arXiv preprint arXiv:2010.06477*, 2020.

[56] 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.

[57] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022.

[58] Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and Qiang Liu. Diffusion-based molecule generation with informative prior bridges. *arXiv preprint arXiv:2209.00865*, 2022.

[59] Muhammed Shuaibi, Adeesh Kolluru, Abhishek Das, Aditya Grover, Anuroop Sriram, Zachary Ulissi, and C Lawrence Zitnick. Rotation invariant graph neural networks using spin convolutions. *arXiv preprint arXiv:2106.09575*, 2021.

[60] Chia-Tche Chang, Bastien Gorissen, and Samuel Melchior. Fast oriented bounding box optimization on the rotation group so (3, r). *ACM Transactions on Graphics (TOG)*, 30(5):1–16, 2011.

[61] Jerome Eberhardt, Diogo Santos-Martins, Andreas F Tillack, and Stefano Forli. Autodock vina 1.2. 0: New docking methods, expanded force field, and python bindings. *Journal of Chemical Information and Modeling*, 61(8):3891–3898, 2021.

[62] Alexey Onufriev, Donald Bashford, and David A Case. Exploring protein native states and large-scale conformational changes with a modified generalized born model. *Proteins: Structure, Function, and Bioinformatics*, 55(2):383–394, 2004.

[63] Yong Duan, Chun Wu, Shibasish Chowdhury, Mathew C Lee, Guoming Xiong, Wei Zhang, Rong Yang, Piotr Cieplak, Ray Luo, Taisung Lee, et al. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. *Journal of computational chemistry*, 24(16):1999–2012, 2003.

[64] Araz Jakalian, Bruce L Bush, David B Jack, and Christopher I Bayly. Fast, efficient generation of high-quality atomic charges. am1-bcc model: I. method. *Journal of computational chemistry*, 21(2):132–146, 2000.

[65] Harrison Green and Jacob D Durrant. Deepfrag: An open-source browser app for deep-learning lead optimization. *Journal of chemical information and modeling*, 61(6):2523–2529, 2021.

A VD-GEN DETAILS

Table 3: Symbol in `VD-Gen`.

Symbol	Meaning
$\mathbf{P}$	the set of atoms in the pocket
$\mathbf{V}_r$	the set of virtual particles (VPs) that are generated at the r-th round in *Equilibrium*
$\mathbf{W}_r$	the set of virtual particles (VPs) that are generated at the r-th round in *Refinement*
$\mathbf{G}$	the set of ground-truth atoms
$\boldsymbol{x}_i^p$	the i-th pocket atom's type (one-hot)
$\boldsymbol{y}_i^p$	the i-th pocket atom's coordinate
$\boldsymbol{x}_i^g$	the i-th ground-truth atom's type (one-hot)
$\boldsymbol{y}_i^g$	the i-th ground-truth atom's coordinate
$\boldsymbol{x}_i^r$	the i-th VP's type (one-hot) at the r-th round in *Equilibrium*
$\boldsymbol{y}_i^r$	the i-th VP's coordinate at the r-th round in *Equilibrium*
$\bar{\boldsymbol{x}}_i^r$	predicted atom type distribution of i-th VP at the r-th round
a_i	The index of assigned target atom for the i-th VP
$\boldsymbol{d}_{ij}^r$	the predicted distance of the i-th and j-th VP pair at the r-th round
$\boldsymbol{d}_{a_i,a_j}^g$	the ground-truth distance of the i-th and j-th VP pair
$\boldsymbol{c}_{ij}^r$	the predicted (ground-truth) distance of the i-th VP and the j-th pocket atom at the r-th round
$\boldsymbol{c}_{a_i,j}^g$	the ground-truth distance of the i-th VP and the j-th pocket atom.
$\hat{\boldsymbol{x}}_i^r$	the i-th VP's type (one-hot) at the r-th round in *Refinement*
$\hat{\boldsymbol{y}}_i^r$	the i-th VP's coordinate at the r-th round in in *Refinement*
$\boldsymbol{q}$	the pair representation of VP pair
$\boldsymbol{q}^l$	the pair representation of VP pair at l-th layer
$\boldsymbol{q}^P$	the pair representation of pocket atom pair
$\boldsymbol{q}^C$	the pair representation of VP and pocket pair
$\bar{\boldsymbol{s}}$	the predicted probability distribution of "success or" not for VP
$\bar{\boldsymbol{r}}$	predicted probability of merging type of VP pair
$\bar{n}$	the predicted atom number
$\boldsymbol{w}_i$	The indices of VPs in the i-th cluster in *Extraction*
$\boldsymbol{h}$	the node representation of VP
$\boldsymbol{h}^l$	the node representation of VP at l-th layer
$\boldsymbol{h}^V$	the node representation of VP in *Equilibrium*
$\boldsymbol{q}^V$	the pair representation of VP pair in *Equilibrium*
$\boldsymbol{h}^W$	the node representation of VP in *Refinement*
$\boldsymbol{q}^W$	the pair representation of VP pair in *Refinement*
$\boldsymbol{h}^P$	the node representation of pocket atom
$\boldsymbol{\theta}_{eq}$	the model parameter in *Equilibrium*
$\boldsymbol{\theta}_{ex}$	the model parameter in *Extraction*
$\boldsymbol{\theta}_{re}$	the model parameter in *Refinement*
$\boldsymbol{\theta}_{co}$	the model parameter in *Confidence*
f	the SE(3) backbone model, return VP types and coordinates
L	the number of layers

A.1 VIRTUAL DYNAMICS

A.1.1 DETAILS OF THE BACKBONE MODEL

In Fig 2 we show the structure of our backbone model, "Repr.", "Attn." and "Dist." are the abbreviations of "Representation", "Attention" and "Distance", respectively. On the left is the pocket encoder, which first uses an atom-type embedding to encode the pocket atom type and a Gaussian kernel to encode the pair-wise distances between pocket atom pairs. In each layer of the pocket encoder, a self-attention layer is used. On the right is the VP encoder, which also uses an atom-type embedding and a Gaussian kernel to encode the particle type and the pair-wise distances between VPs. To interact with the pocket encoder, another Gaussian kernel is used to encode the pair-wise distances between VPs and pocket atoms. In each layer of the VP encoder, before the self-attention layer, a particle-pocket attention layer is used to interact with the pocket encoder.

We describe the components in the backbone model in the following paragraphs. Besides, we also describe the overall pipeline of the backbone model in the Alg. 2. For simplicity, layer normalization is not shown in the equations and algorithms.

Gaussian kernel The pair-type aware Gaussian kernel [59; 28] is denoted as:

$$\boldsymbol{p}_{ij} = \{\mathcal{G}(\mathcal{A}(d_{ij}, t_{ij}; \boldsymbol{a}, \boldsymbol{b}), \mu^k, \sigma^k) | k \in [1, D]\}, \quad \mathcal{A}(d, r; \boldsymbol{a}, \boldsymbol{b}) = a_r d + b_r, \tag{2}$$

where $\mathcal{G}(d, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(d-\mu)^2}{2\sigma^2}}$ is a Gaussian density function with parameters μ and σ, d_{ij} is the Euclidean distance of atom pair ij, and t_{ij} is the pair-type of atom pair ij. $\mathcal{A}(d_{ij}, t_{ij}; \boldsymbol{a}, \boldsymbol{b})$ is the affine transformation with parameters $\boldsymbol{a}$ and $\boldsymbol{b}$, it affines d_{ij} corresponding to its pair-type t_{ij}.

Pair representation Pair representation [28] is used to further enhance the 3D spatial encoding. The update of pair representation is via the multi-head Query-Key product results in self-attention.

$$\boldsymbol{q}_{ij}^{l+1} = \boldsymbol{q}_{ij}^{l} + \{\frac{\boldsymbol{h}_i^l \boldsymbol{W}_{l,h}^Q (\boldsymbol{h}_j^l \boldsymbol{W}_{l,h}^K)^T}{\sqrt{d}} | h \in [1, H]\}, \tag{3}$$

where $\boldsymbol{h}_i^l$ is the atom/node representation of the i-th atom at l-th layer, $\boldsymbol{q}_{ij}^l$ is the pair representation of atom pair ij in l-th layer, H is the number of attention heads, d is the dimension of hidden representations, and $\boldsymbol{W}_{l,h}^Q$ ($\boldsymbol{W}_{l,h}^K$) is the projection for Query (Key) of the l-th layer h-th head.

To leverage 3D information in the atom representation, pair representation is used in self-attention.

$$\begin{aligned} \boldsymbol{h}_i^{l+1,h} &= \text{softmax}(\frac{\boldsymbol{h}_i^l \boldsymbol{W}_{l,h}^Q (\boldsymbol{h}_j^l \boldsymbol{W}_{l,h}^K)^T}{\sqrt{d}} + \boldsymbol{q}_{ij}^{l,h}) \boldsymbol{h}_j^l \boldsymbol{W}_{l,h}^V, \\ \boldsymbol{h}_i^{l+1} &= \text{concat}_h(\boldsymbol{h}_i^{l+1,h}), \end{aligned} \tag{4}$$

where $\boldsymbol{W}_{l,h}^V$ is the projection of Value of the l-th layer h-th head.

Particle-Pocket Attention The Particle-Pocket Attention can be denoted as the following:

$$\begin{aligned} \boldsymbol{h}_i^{l+1,h} &= \text{softmax}(\frac{\boldsymbol{h}_i^l \boldsymbol{W}_{l,h}^{P,Q} (\boldsymbol{h}_j^P \boldsymbol{W}_{l,h}^{P,K})^T}{\sqrt{d}} + \boldsymbol{q}_{ij}^{C,l,h}) \boldsymbol{h}_j^P \boldsymbol{W}_{l,h}^{P,V}, \\ \boldsymbol{h}_i^{l+1} &= \text{concat}_h(\boldsymbol{h}_i^{l+1,h}), \\ \boldsymbol{h}_i^{l+1} &= \boldsymbol{h}_i^l + g_1 \cdot \boldsymbol{h}_i^{l+1} + g_2 \cdot \text{MLP}(\boldsymbol{h}_i^{l+1}), \end{aligned} \tag{5}$$

where g_1 and g_2 are learned parameters with initialized value 0, $\boldsymbol{h}_j^P$ is the representation of the j-th pocket atom, $\boldsymbol{q}_{ij}^{C,l,h}$ is the pair representation of particle-pocket pair ij in l-th layer h-th head, MLP is a full-connected network with one hidden layer. $\boldsymbol{W}_{l,h}^{P,Q}$, $\boldsymbol{W}_{l,h}^{P,K}$, and $\boldsymbol{W}_{l,h}^{P,V}$ are learnable projections for Query, Key and Value.

SE(3)-equivariance coordinate Following [28], the head could be denoted as:

$$\boldsymbol{y}_i^{r+1} = \boldsymbol{y}_i^r + \sum_{j=1}^{n} \frac{(\boldsymbol{y}_i^r - \boldsymbol{y}_j^r) z_{ij}}{n}, \quad z_{ij} = \text{ReLU}((\boldsymbol{q}_{ij}^L - \boldsymbol{q}_{ij}^0)\boldsymbol{U}_1)\boldsymbol{U}_2, \tag{6}$$

where n is the number of total atoms, L is the number of layers in model, $\boldsymbol{y}_i^r \in \mathbb{R}^3$ is the input coordinate of i-th atom, and $\boldsymbol{y}_i^{r+1} \in \mathbb{R}^3$ is the output coordinate of i-th atom, $\boldsymbol{U}_1 \in \mathbb{R}^{H \times H}$ and $\boldsymbol{U}_2 \in \mathbb{R}^{H \times 1}$ are the projection matrices to convert pair representation to scalar.

Atom Type Prediction Head We use a non-linear head with two layers to predict the atom type based on the atom representation in the last layer of the particle encoder:

$$\bar{\boldsymbol{x}}_i = \text{MLP}(\boldsymbol{h}_i^L) \tag{7}$$

where $\boldsymbol{h}_i^L$ is the atom representation, L is the number of layers of the particle encoder,

Algorithm 2 Backbone_Update

Require: $\mathbf{P}$: pocket atoms, $\mathbf{V}_r$: virtual particles

$\boldsymbol{h}^{P,0} \leftarrow$ Atom_Type_Embedding$(\mathbf{P})$ ▷ Embeddings from atom types
$\boldsymbol{q}^{P,0} \leftarrow$ Gaussian_Kernel(Dist_Matrix$(\mathbf{P}, \mathbf{P})$) ▷ Get invariant spatial positional embedding
for $l \in [1, ..., L)$ **do** ▷ Update Pocket Encoder
  $\boldsymbol{h}^{P,l}, \boldsymbol{q}^{P,l} \leftarrow$ Self_Attn$(\boldsymbol{h}^{P,l-1}, \boldsymbol{q}^{P,l-1})$) ▷ Update by self attention
  $\boldsymbol{h}^{P,l} \leftarrow$ MLP$(\boldsymbol{h}^{P,l})$ ▷ Update by Feed-Forward-Network
$\boldsymbol{h}^{P} \leftarrow \boldsymbol{h}^{P,L}$
$\boldsymbol{h}^{0} \leftarrow$ Atom_Type_Embedding$(\mathbf{V}_r)$ ▷ Embeddings from atom types
$\boldsymbol{q}^{0} \leftarrow$ Gaussian_Kernel(Dist_Matrix$(\mathbf{V}_0, \mathbf{V}_0)$) ▷ Get invariant spatial positional embedding
$\boldsymbol{q}^{C,0} \leftarrow$ Gaussian_Kernel(Dist_Matrix$(\mathbf{V}_0, \mathbf{P})$) ▷ Get invariant spatial positional embedding of particle-pocket pairs
for $l \in [1, ..., L)$ **do** ▷ Update Particle Encoder
  $\boldsymbol{h}^{l}, \boldsymbol{q}^{l} \leftarrow$ Self_Attn$(\boldsymbol{h}^{l-1}, \boldsymbol{q}^{l-1})$) ▷ Update by self attention
  $\boldsymbol{h}^{l} \leftarrow$ MLP$(\boldsymbol{h}^{l})$ ▷ Update by Feed-Forward-Network
  if $l \mod 4 == 0$ **then** ▷ Only enabled at every 4-layer
    $\boldsymbol{h}^{l}, \boldsymbol{q}^{C,l} \leftarrow$ Particle_Pocket_Attn$(\boldsymbol{h}^{l}, \boldsymbol{h}^{P}, \boldsymbol{q}^{C,l-1})$) ▷ Update by Particle-Pocket Attention
$\bar{\boldsymbol{x}}^{r+1} \leftarrow$ Atom_Type_Head$(\boldsymbol{h}^{L})$ ▷ Atom Type Prediction
$\boldsymbol{x}^{r+1} \leftarrow$ sample$(\bar{\boldsymbol{x}}^{r+1})$ ▷ Sample an atom type based on predicted probability
$\boldsymbol{y}^{r+1} \leftarrow$ SE(3)_Head$(\boldsymbol{y}^{r}, \boldsymbol{q}^{L})$ ▷ Coordinate update
return $\mathbf{V}_{r+1} = \{\boldsymbol{x}^{r+1}, \boldsymbol{y}^{r+1}\}, \boldsymbol{h}^{L}, \boldsymbol{q}^{L}, \boldsymbol{h}^{P}$

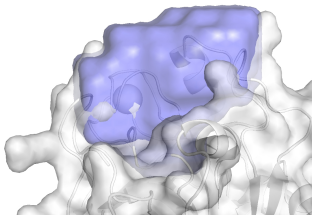


Figure 5: A case to show the detected pocket cavity.

A.1.2 Pocket Cavity Discovery

Pocket cavity discovery is an essential component in VD-Gen, as VPs need to scatter into the cavity. To find the pocket cavity, we first use OBB (oriented bounding box) [60] to determine a cubic box, denote as $\mathcal{B}$, based on the pocket's residue atoms. Then, we enlarge the box a little bit, increased by 4 Å. Then, we make the 3D grids with resolution 2 Å, for the whole protein, including the pocket, and mark the grids that contain protein atoms as "used". Then, starting from a given grid inside the cavity, a breadth-first search is used to find the grids inside the cavity. In particular, the grids marked as "used" or are not in $\mathcal{B}$ are not considered. We show an example in Fig 5, where the purple region indicates the pocket cavity we find.

A.2 Extraction

Filtering Loss

$$\mathcal{L}_{Filter} = \frac{1}{n}\sum_{i=1}^{n} \text{NLL}(\bar{\boldsymbol{s}}_i, \boldsymbol{s}_i) \tag{8}$$

where n is the number of virtual particles, $\bar{\boldsymbol{s}}_i$ is the predicted probability distribution of success or not, $\boldsymbol{s}_i$ is the target label. The target label is marked as "success" if the distances between VPs and their target positions after Equilibrium are smaller than 2.5 Å.

Merging Loss

$$\mathcal{L}_{Merge} = \frac{1}{l^2}\sum_{i=1}^{l}\sum_{j=1}^{l}\sum_{k=1}^{2} -\boldsymbol{r}_{ij}^{k} \log \bar{\boldsymbol{r}}_{ij}^{k} \alpha_k (1-\bar{\boldsymbol{r}}_{ij}^{k})^{\gamma}, \tag{9}$$

where l is the number of virtual particles predicted to be "success", $\boldsymbol{r}_{ij}$ is the target merging type, $\bar{\boldsymbol{r}}_{ij}$ is the predicted probability of merging type, the blue part is from focal loss [32], and α_k and γ are

hyper-parameters to balance classes. Here γ is set to 2, the subscript 1 of α represents the True type and α_1 is set 10 while α_0 is set to 1.

Atom number loss For atom number prediction, we bucket the number of atoms into different bins and transform the numerical problem into a classification problem to make the training more stable.

$$\mathcal{L}_{atom\_num} = \text{NLL}(\bar{\boldsymbol{o}}, \boldsymbol{o}) \tag{10}$$

where $\bar{\boldsymbol{o}}$ is the predicted probability distribution of bins, and $\boldsymbol{o}$ represents the one-hot vector of the target bin.

During inference, the predicted atomic number can be calculated from the predicted distribution over bins.

$$\sum_{k=1}^{n_{bin}} (bin\_val_k)\bar{\boldsymbol{o}}_k, \tag{11}$$

where bin_val_k is the bin value of the k-th bin, n_{bin} is the number of bins, l is the size of each bin and $\bar{o}_k$ is the predicted probability of the k-th bin. Notably, the bin value is not the bin boundary value, it is the average of left and right boundaries.

Merge algorithm The detail of merging VPs into atoms are shown in Alg 3. In particular, a binary search is used to find a merging threshold. During training, teacher-forcing merging is used for reducing the training cost (without binary search). This is, rather than predicting pair-wise merge probabilities and the atom number, we directly used their ground truth values. During inference, the binary search is used. Besides, considering the error in atom number prediction, we try a range (± 10) of atom numbers, and select from them based on their confidence scores.

A.3 CONFIDENCE

The confidence score is based on LDDT metric [33]:

$$\text{LDDT} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j\neq i}\frac{1}{4}((\text{err}_{ij} < 0.5) + (\text{err}_{ij} < 1.0) + (\text{err}_{ij} < 2.0) + (\text{err}_{ij}) < 4.0), \tag{12}$$

$$\text{err}_{ij} = \text{L1}(||\hat{\boldsymbol{y}}_i^r - \hat{\boldsymbol{y}}_j^r||_2, ||\hat{\boldsymbol{y}}_i^g - \hat{\boldsymbol{y}}_j^g||_2), \tag{13}$$

where $\hat{\boldsymbol{y}}_i^r$ is the predicted coordinate of i-th particle after *Refinement*, and $\hat{\boldsymbol{y}}_i^g$ is its ground truth coordinate. Then, a task is trained to predict the LDDT score. Here, we use the binning trick for training stability, bucketing the error of each VP into different bins, and using cross-entropy loss to train the task:

$$\mathcal{L}_{Confidence} = \frac{1}{n}\sum_{i=1}^{n}\text{NLL}(\bar{\boldsymbol{e}}_i, \boldsymbol{e}_i) \tag{14}$$

where $\bar{\boldsymbol{e}}_i$ is the predicted error's probability distribution of i-th VP and $\boldsymbol{e}_i$ represents the one-hot vector of the real error.

A.4 PRETRAIN

During pretraining, we remove atoms in a continuous spatial region, and VPs are initialized in the region. To find the atoms in a continuous region, we design a greedy algorithm. First, we initialize an empty atom set, then we randomly select an atom in the molecular to join the set. Starting from the atom set, we choose an atom closest to the atoms in the set and join it to the set. We repeat the process until the number of atoms in the set meets the requirements. And we use OBB (oriented bounding box) [60] to determine a cubic box by the removed atoms, and VPs are initialized inside the cubic box.

A.5 VD-GEN OVERALL ALGORITHM

We also summarize the overall inference pipeline of `VD-Gen` in the Alg. 4. The algorithm mainly relies on the function "VD", which iteratively moves the VPs. Both *Equilibrium* and *Refinement* use

Algorithm 3 Filter_Merge_VPs

Require: $\mathbf{V}_R = \{(\boldsymbol{x}_i^R, \boldsymbol{y}_i^R)\}_{i=1}^n$: virtual particles at R-th rounds, $\bar{n}$: the predicted atom num, $\bar{\boldsymbol{r}} = \{(\bar{\boldsymbol{r}}_{ij})\}_{i=1,j=1}^{n\times n}$: the predicted merging probability matrix, $\bar{\boldsymbol{s}} = \{\bar{\boldsymbol{s}}_i\}_{i=1}^n$: the predicted filtering probability

for i in $[1, ..., n]$ **do** ▷ Set the merge type between particles with the particle to be filtered to 0
  $s_i \leftarrow \text{argmax}(\bar{\boldsymbol{s}}_i)$ ▷ Get the filtering type
  if $s_i = 0$ **then** ▷ If the i-th particle should be filtered
    $\bar{\boldsymbol{r}}[:, i] \leftarrow 0$ ▷ Set $\{\bar{\boldsymbol{r}}_{i,j}\}_{j=1}^n$ to 0
    $\bar{\boldsymbol{r}}[i, :] \leftarrow 0$ ▷ Set $\{\bar{\boldsymbol{r}}_{j,i}\}_{j=1}^n$ to 0
high $\leftarrow \max(\{(\bar{\boldsymbol{r}}_{ij})\}_{i=1,j=1}^{n\times n})$
low $\leftarrow \min(\{(\bar{\boldsymbol{r}}_{ij})\}_{i=1,j=1}^{n\times n})$
mid $\leftarrow \frac{\text{low}+\text{high}}{2}$
while low $<$ high **do** ▷ using the binary search to find the threshold
  $\boldsymbol{r}_{ij} \leftarrow \bar{\boldsymbol{r}}_{ij} >$ mid
  $\mathbf{W}_0 \leftarrow [], \boldsymbol{w} \leftarrow [], m \leftarrow 0$
  for i in **random_perm**$(1, n)$ **do** ▷ Greedy merge based a random order
    $\boldsymbol{w}_i \leftarrow []$,
    for j in $[1, ..., n]$ **do**
      if $\boldsymbol{r}_{ij} =$ True **then** ▷ the j-th particle should be merged
        $\boldsymbol{r}[:, j] \leftarrow$ False ▷ the merged particle will not be merged again
        $\boldsymbol{w}_i$.add(j) ▷ add the particle indices into the i-th cluster
    if len$(\boldsymbol{w}_i) > 0$ **then**
      $\hat{\boldsymbol{x}}_m^0 \leftarrow \text{Uniform}(\{\boldsymbol{x}_k^r | k \in \boldsymbol{w}_i\})$ ▷ Sample atom type from the merging list
      $\hat{\boldsymbol{y}}_m^0 \leftarrow \text{Mean}(\{\boldsymbol{y}_k^r | k \in \boldsymbol{w}_i\})$ ▷ the average position is atom position after merging
      $\mathbf{W}_0$.add$((\hat{\boldsymbol{x}}_m^0, \hat{\boldsymbol{y}}_m^0))$ ▷ add the atom to the set
      $m \leftarrow m + 1$ ▷ Count the number of clusters
  if $m = \bar{n}$ **then** ▷ find the threshold
    break
  else
    if $m < \bar{n}$ **then** ▷ too many particles are merged, the threshold needs to be increased
      low $\leftarrow$ mid
    else
      high $\leftarrow$ mid ▷ Too few particles are merged, the threshold needs to be lowered
return $\mathbf{W}_0$ ▷ Return particle set after merging

"VD". In *Extraction*, several heads are used to predict the filtered probability, the pair merge probability, and the number of atoms of the ligand molecule. Based on these predictions, "Filter_Merge_VPs" is used to extract the merged VPs.

The training pipeline is very similar, except for the following differences:

- For efficiency purposes, R_1 and R_2 are sampled during training, and the gradient backward is only enabled in the last round.
- For efficiency purposes, in "Filter_Merge_VPs", teacher-forcing merging (without binary search) is used during training. This is, rather than predicting pair-wise merge probabilities and the atom number, we directly used their ground truth values.
- The loss functions are enabled to get gradients to train models.

B Experiment details and more results

B.1 Training details

The detailed configurations of `VD-Gen` are listed in Table 4. We did not tune these hyper-parameters for now, a better performance could be achieved with well-tuned hyper-parameters.

B.2 Evaluation Mertic

- *3D similarity*. We use LIGSIFT [25] to calculate 3D similarity. However, by default, LIGSIFT will align the input molecules before calculating 3D similarity. But we want to evaluate the generated

Algorithm 4 VD-Gen Framework during Inference

Require: R_1, R_2: max rounds in *Equilibrium* and *Refinement*, $\mathbf{P}$: pocket atoms with types and positions, n: number of VPs, $\boldsymbol{\theta}_{eq}$, $\boldsymbol{\theta}_{ex}$, $\boldsymbol{\theta}_{re}$, $\boldsymbol{\theta}_{co}$: model parameters in different stages

Virtual dynamics mechanism
def VD(R, $\mathbf{V}_0$, $\mathbf{P}$, $\boldsymbol{\theta}$): ▷ Virtual dynamics mechanism
  for $k \in [1, ..., R]$ **do** ▷ Iterative updates without gradients
    $\mathbf{V}_k, \boldsymbol{h}^L, \boldsymbol{q}^L, \boldsymbol{h}^P \leftarrow$ Backbone_Update($\mathbf{V}_{k-1}, \mathbf{P}; \boldsymbol{\theta}$) ▷ backbone model update as in Alg. 2
  return $\mathbf{V}_R, \boldsymbol{h}^L, \boldsymbol{q}^L, \boldsymbol{h}^P$

Initialize VPs
$\tilde{P} =$ Cavity_Discovery($\mathbf{P}$) ▷ Get the pocket cavity as in Appendix A.1.2
for i in [1,...,n] **do**
  $\boldsymbol{x}_i^0 \leftarrow$ one_hot([MASK]) ▷ The types of VPs are initialized as a meaningless [MASK] type
  $\boldsymbol{y}_i^0 \leftarrow$ Uniform($\tilde{P}$) ▷ The initial coordinates of VPs are uniformly sampled from the cavity
$\mathbf{V}_0 = \{(\boldsymbol{x}_i^0, \boldsymbol{y}_i^0)\}_{i=1}^n$

Equilibrium stage
$\mathbf{V}_R, \boldsymbol{h}^V, \boldsymbol{q}^V, \boldsymbol{h}^P \leftarrow \mathrm{VD}(R_1, \mathbf{V}_0, \mathbf{P}, \boldsymbol{\theta}_{eq})$ ▷ Predict coordinates and types with VD

Extraction stage
$\bar{\boldsymbol{s}} \leftarrow$ Filter_Head($\boldsymbol{h}^V; \boldsymbol{\theta}_{ex}$) ▷ Predict to filter the "not success" particles
$\bar{\boldsymbol{r}} \leftarrow$ Merge_Head($\boldsymbol{q}^V; \boldsymbol{\theta}_{ex}$) ▷ Predict merging matrix
$\bar{n} \leftarrow$ Atom_Num_Head($\boldsymbol{h}^P; \boldsymbol{\theta}_{ex}$) ▷ Predict the atom number
$\mathbf{W}_0 \leftarrow$ Filter_Merge_VPs($\mathbf{V}_R, \bar{n}, \bar{\boldsymbol{r}}, \bar{\boldsymbol{s}}$)
  ▷ Filter and Merge VPs as in Alg. 3 using predicted merging matrix, atom number and filtering type

Refinement stage
$\mathbf{W}_R, \boldsymbol{h}^W, \boldsymbol{q}^W, \boldsymbol{h}^P \leftarrow \mathrm{VD}(R_2, \mathbf{W}_0, \mathbf{P}; \boldsymbol{\theta}_{re})$ ▷ Refine the coordinates and types
Confidence stage
Pred_LDDT $\leftarrow$ Confidence_Head($\boldsymbol{h}^W, \boldsymbol{\theta}_{co}$) ▷ Predict LDDT

return $\mathbf{W}_R$, Pred_LDDT ▷ Return the final positions and types, and the confidence score

3D structure directly, to examine the end-to-end performance. Therefore, we *remove the alignment in LIGSIFT*.

- *Vina*. We use AutoDock Vina1.2 [61] to get Vina score. In particular, the re-docking will be applied. That is, the binding pose and the conformation of the ligand molecule generated by the model will be ignored, and a new binding pose and a new molecular conformation will be re-calculated by AutoDock Vina1.2. We believe the re-docking in Vina cannot reflect the actual performance of the pocket-based 3D molecular generation. But to be consistent with previous works, we still use it as one of the metrics.

- *Vina**. Vina* is Vina without re-docking. In particular, we use the built-in energy optimization process based on Vina scoring function in AutoDock Vina1.2 [61] to minimize the energy of the binding pose of generated molecules, and then use the Vina scoring function to score the energy-minimized binding pose to get Vina* score.

- *MM-PBSA*. We take the default settings of parameters (i.e., solvation mode: GB-2[62], protein forcefield: amber03[63], ligand charge method: bcc[64], dielectric constant: 4.0) and workflow (i.e., force field building, structure optimization by energy minimization, MM/GB(PB)SA calculation) of [24] to calculate MM-PBSA score. Since the crystal structure indicates the preferred binding pose against a specific target, we filtered the generated molecules by 3D similarity to the molecule in crystal structure and take the molecules whose 3D similarity score is over 0.4 as effective molecules, and we only calculate the MM-PBSA score for the effective molecules. In Table 6 we show MM-PBSA S.R. (success rate), which calculates the proportion of effective MM-PBSA of

the generated molecules. For MM-PBSA B.T. and MM-PBSA Rank we have:

$$\text{MM-PBSA B.T.} = \frac{1}{n_p}\sum_{i=1}^{n_p}\frac{|\{g \in \mathcal{G}|\text{MM-PBSA}(g) < \text{MM-PBSA}(\overline{m}_i)\}|}{|\mathcal{G}|}, \tag{15}$$

$$\text{MM-PBSA Rank} = \frac{1}{n_p}\sum_{i=1}^{n_p}\text{rank}_i, \tag{16}$$

where n_p is the number of proteins in the test set, $\mathcal{G}$ represents the generated molecular set, $\overline{m}_i$ represents the molecular in the crystal structure of the i-th protein and rank_i represents the ranking index of the current model among all of the compared models under the i-th protein which is ranked by MM-PBSA.

- Metric for ablation studies. We use 3D similarity between the generated molecules and the ground truth as the metric in ablation studies since it reflects the generative ability based on the pocket structure and there is a strong correlation between 3D similarity and binding affinity according to Table 1.

Table 4: Settings for `VD-Gen`.

Pretrain	
VP encoder layers	12
Peak learning rate	1e-4
Batch size	128
Max training steps	1M
Warmup steps	10K
Attention heads	64
FFN dropout	0.1
Attention dropout	0.1
Embedding dropout	0.1
Weight decay	1e-4
Embedding dim	512
FFN hidden dim	2048
Gaussian kernel channels	128
Activation function	GELU
Learning rate decay	Linear
Adams ϵ	1e-6
Adams(β_1, β_2)	(0.9,0.99)
Gradient clip norm	1.0
Particle type prediction weight	1.0
Loss weight for $\mathcal{L}_{\mathcal{VD}}$ of *Equilibrium*	1.0
Loss weight for $\mathcal{L}_{\mathcal{VD}}$ of *Refinement*	1.0
Loss weight for $\mathcal{L}_{Filter}$	1.0
Loss weight for $\mathcal{L}_{Merge}$	5.0
Loss weight for $\mathcal{L}_{atom\_num}$	1.0
Loss weight for *Confidence*	1.0
R, max round of iterative movement of *Equilibrium* and *Refinement*	4
numbers of virtual particles	$8 \sim 9$ times of the number of real atoms
τ, the clip value for coordinate loss	2
δ, the threshold for coordinate regularization	1
Finetune	
Batch size	64
Max training steps	100K
R_1, max round of iterative movement of *Equilibrium*	4
R_2, max round of iterative movement of *Refinement*	4
numbers of virtual particles	$16 \sim 18$ times of the number of real atoms
Inference	
R_1, max round of iterative movement of *Equilibrium*	4
R_2, max round of iterative movement of *Refinement*	16
n, numbers of VPs	512

B.3 More Results

In Table 5, we report more percentile results for Vina, Vina*. In Table 6, we report more percentile MM-PBSA results and MM-PBSA S.R. scores. The MM-PBSA S.R. scores in many baselines are

very low. Thus, there are not enough effective MM-PBSA results to calculate percentile results in some baselines. Therefore, in each pocket, we replace the failed MM-PBSA result with the worst one generated by that baseline. And we calculated the percentile results after the replacement.

Table 5: More results on Vina and Vina*.

Model	5-th		10-th		25-th		50-th	
	Vina(↓)	Vina*(↓)	Vina(↓)	Vina*(↓)	Vina(↓)	Vina*(↓)	Vina(↓)	Vina*(↓)
LiGAN[17]	-6.724	-5.372	-6.324	-4.922	-5.740	-4.215	-5.065	-3.49
3DSBDD[18]	-8.662	-7.227	-8.296	-6.664	-7.557	-5.633	-6.474	-4.078
GraphBP[19]	-8.710	-3.689	-7.832	-2.774	-6.765	-1.169	-5.625	-1.2
Pocket2Mol[20]	-8.332	-6.525	-8.015	-5.399	-7.467	-3.513	-6.837	-1.808
VD-Gen	**-9.047**	**-7.444**	**-8.652**	**-6.848**	**-7.958**	**-5.825**	**-7.146**	**-4.621**

Table 6: More MM-PBSA results.

Model	5-th MM-PBSA(↓)	10-th MM-PBSA(↓)	25-th MM-PBSA(↓)	50-th MM-PBSA (↓)	MM-PBSA-S.R.(%↑)
LiGAN[17]	-17.865	-13.374	-8.775	-7.418	11.9
3DSBDD[18]	-30.221	-23.623	-13.544	-7.739	12.9
GraphBP[19]	-5.130	-4.894	-4.894	-4.894	0.2
Pocket2Mol[20]	-7.823	-5.945	-5.398	-5.398	1.8
VD-Gen	**-51.258**	**-47.247**	**-39.984**	**-21.140**	**42.7**

B.4 Compared with Image Generation and Some Early Attempts

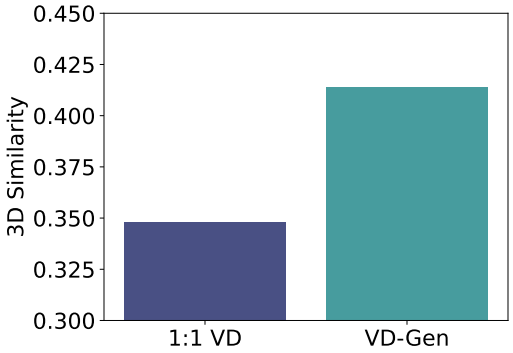


Figure 6: Comparison for different training frameworks. "1:1 VD" is our early attempt, which is directly based on Virtual Dynamics, with 1:1 particle-atom assignment, and without the 4 stages in VD-Gen. The result indicates the effectiveness of the proposed VD-Gen framework.

During inference, VD iteratively moves particles to more precious positions from random initialized positions. A similar idea of "coarse-to-fine" generation is widely used in image generative models, and images could be iteratively refined from noises.

From this view, VD looks similar to the "coarse-to-fine" image generation. However, the training of VD is more challenging and very different from "coarse-to-fine" image generation.

- In image generation, the training target for each pixel is straightforward to assign, since input pixels' positions are the same as the ground-truth pixels' position, i.e. there is a 1-to-1 mapping between input and ground-truth. For example, in an image with 32x32 pixels, for an input pixel located at position i, j, we can directly use the ground-truth pixel located at position i, j as its training target. With the 1-to-1 assignment, the training of image generation is straightforward.
- However, in the 3D molecular generation, the 1-to-1 assignment cannot simply be used, as the randomly initialized 3D positions of input particles are far different from the ground-truth atoms' positions, so it is hard to have 1-to-1 mapping. Besides, the number of ground-truth atoms is unknown, which further increases the difficulty of 3D molecular generation.

Table 7: Inference Efficiency.

Model	3DSBDD	GraphBP	Pocket2Mol	VD-Gen
Time(s)(↓)	14.153	1.660	3.476	3.678

- In our early attempts, we also tried some 1-to-1 assignment methods (assuming the ground truth of the number of atoms is given). We first tried the random assignment, and found model training is hard to converge. We then optimized it by considering the total moving distance of all input particles in the target assignment. We call this method "1:1 VD", details are in the following paragraph. In particular, we assign each particle a unique target atom, by sub-optimal assignment (optimal assignment is NP-hard) to minimize the total moving distance. It is much better than random assignment, but the performance is still not good, and we think the reason is due to the large difficulty of the training task.
- To further reduce the difficulty of the training task, we propose to use the many-to-1 assignment. That is, we first use many random-scattered particles to the pocket cavity, then for each particle, assign its nearest ground-truth atom as the training target. With this solution, the learning of movement is much easier, since particles only consider their nearest atoms. Besides, the unknown atom number is not a problem.
- Besides, VD-Gen is not just VD. Simply using VD, we can only get a 3D density-like shape (formed by the positions of particles) of a 3D molecule. We may use some rule-based solutions, like clustering by distances, to extract 3D molecules from the shape. However, rule-based solutions are not end-to-end and could fail in various scenarios. To address this, we further propose VD-Gen, a more reliable framework with additional Extraction, Refinement and Confidence stages.

1:1 VD In our early attempt, we tried a simple solution: use the same number of VPs as real atoms, and randomly scatter them; then, make an assignment so that each atom has a paired VP, and each VP has a paired atom. The optimal assignment with minimal moving distance is NP-hard, and we use a greedy algorithm to find a sub-optimal assignment. With the 1-to-1 assignment, the *Extraction* stage is not needed, since the number of VPs is the same as real atoms. We called this method "1:1 VD". For a fair comparison, pretraining is also used in "1:1 VD". We conduct the experiment to compare VD-Gen with "1:1 VD", the results are shown in Fig. 6. From the result, we find that VD-Gen largely outperforms "1:1 VD". Although its simplicity, the learning of "1:1 VD" is challenging, due to the ambiguous target assignment which violates the least action principle. As for VD-Gen, although it looks complicated with multiple stages, these stages are necessary for generating accurate 3D molecules end-to-end.

B.5 Inference Efficiency

Experiment results have demonstrated the effectiveness of the proposed VD-Gen, and we also check its efficiency here. In particular, we benchmark the inference speed of generating one molecule for 3DSDBB, GraphBP, Pocket2Mol, and our VD-Gen. The results are summarized the Table 7. 3DSBDD is the slowest one, due to the inefficient MCMC sampling. Although GraphBP is the fastest one, its generated molecules are the worst. VD-Gen and Pocket2Mol are similar in efficiency. But VD-Gen significantly outperforms Pocket2Mol in effectiveness. Due to the large number of VPs and several movement rounds, it is expected that VD-Gen is not the fastest one. We leave the efficiency improvement to future work.

B.6 Illustration of VPs' movement

We show an example of VPs' spatial position during the inference of VD-Gen in Fig 7. In the initial stage, the coordinates of VPs are randomly initialized. In *Equilibrium* stage, with the increase of movement rounds($1 \sim 4$), VPs gradually gather together. Then in *Extraction* stage, after clustering, fewer VPs are extracted from gathered VPs. Then in *Refinement* stage, the extracted VPs continue the iterative movement, toward positions with better pLDDT scores.

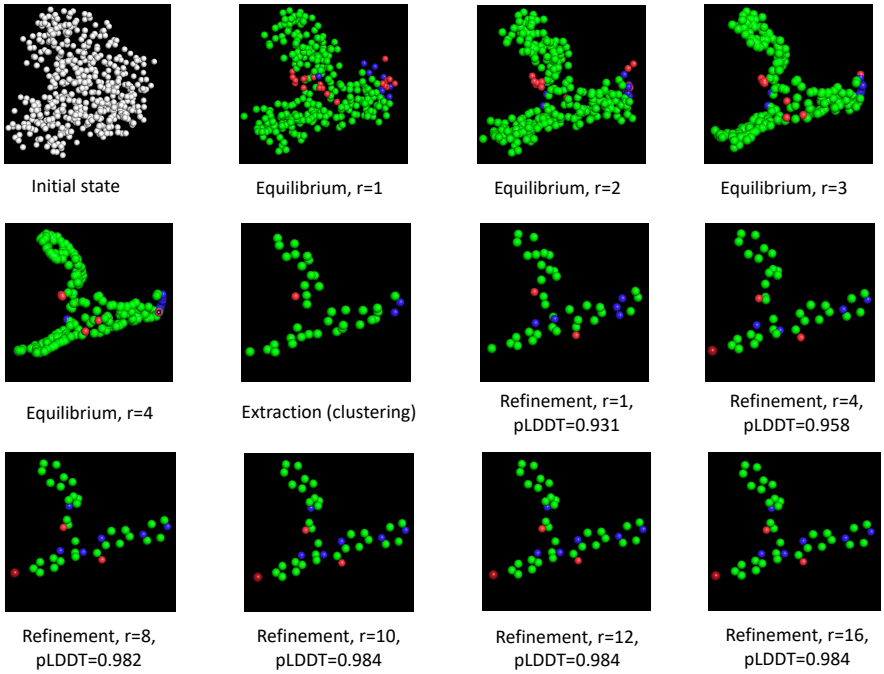


Figure 7: An example to show how the VPs moves at each iteration in *Equilibrium* and *Refinement*, r indicates the moving iterations and pLDDT can reflect the change of coordinates.

Table 8: Training on CrossDocked Dataset.

Model	LiGAN	3DSBDD	GraphBP	Pocket2Mol	VD-Gen
3D Similarity(↑)	0.356	0.365	0.333	0.352	**0.39**

B.7 Training on the CrossDocked dataset

Since the baselines use the cross-docked dataset as training data, to analyze our model effect without pretrain, we conduct experiments on the cross-docked dataset. Results are shown in Table 8. We can see VD-Gen achieves 3D similarity with 0.39, outperforming other baselines. Besides, compared to VD-Gen with particle encoder pertaining in Table 2, the performance of using the cross-docked dataset is worse (0.39 v.s. 0.402). This result also indicates that pretraining is better than data augmentation in the cross-docked dataset.

B.8 Molecular optimization task

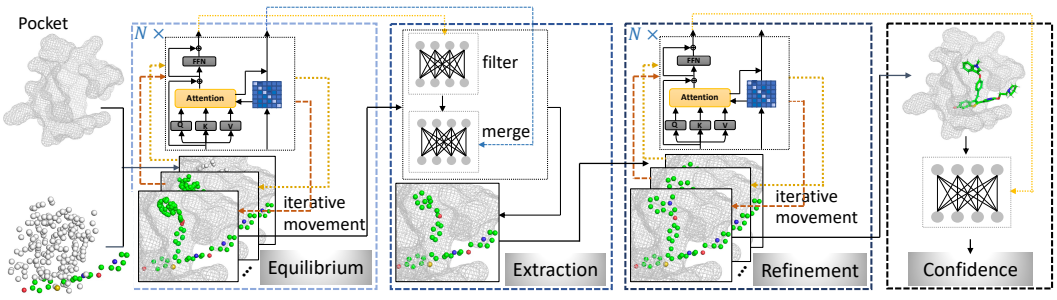


Figure 8: Extending VD-Gen to molecular optimization.

Difference in training molecular optimization models To train the molecular optimization model, we make the following changes.

- The remove ratio in pretraining is much smaller, only 25% to 40% are removed.
- Rather than removing the whole molecule, during finetuning, only 25% to 40% of atoms are removed, like the pretraining.
- During training, the number of VPs is also much smaller, only 8 times of the real atoms.
- The VPs are not scattered in the whole pocket cavity, but scattered around the removed atoms.

Experiment We compare our model with a traditional molecular fragments optimization model DeepFrag [65]. DeepFrag can replace molecular fragments based on SMILES, which is a 1D model without pocket information. The results are shown in Table 9 and Table 10. From them, it is clear that `VD-Gen` can outperform the baseline in molecular optimization.

Table 9: Full percentile results on Vina and Vina*, in molecular optimization tasks.

Model	5-th		10-th		25-th		50-th	
	Vina(↓)	Vina*(↓)	Vina(↓)	Vina*(↓)	Vina(↓)	Vina*(↓)	Vina(↓)	Vina*(↓)
DeepFrag[65]	-8.357	-	-8.132	-	-7.775	-	-7.372	-
`VD-Gen`	**-9.040**	-8.30	**-8.775**	-8.020	**-8.333**	-7.507	**-7.880**	-6.946

Table 10: Full percentile results on MM-PBSA, in molecular optimization tasks.

Model	5-th MM-PBSA(↓)	10-th MM-PBSA(↓)	25-th MM-PBSA(↓)	50-th MM-PBSA (↓)	MM-PBSA B.T.(↑)
DeepFrag[65]	-51.783	-48.959	-39.786	-34.485	23.9
`VD-Gen`	**-53.799**	**-52.120**	**-46.707**	**-41.788**	**38.3**