Towards Predicting Equilibrium Distributions for Molecular Systems with Deep Learning

Shuxin Zheng[1*†], Jiyan He[1†], Chang Liu[1*†], Yu Shi[1†], Ziheng Lu[1†], Weitao Feng[1], Fusong Ju[1], Jiaxi Wang[1], Jianwei Zhu[1], Yaosen Min[1], He Zhang[1], Shidi Tang[1], Hongxia Hao[1], Peiran Jin[1], Chi Chen[2], Frank Noé[1], Haiguang Liu[1*†] and Tie-Yan Liu[1*]

[1]Microsoft Research AI4Science.
[2]Microsoft Quantum.
https://DistributionalGraphormer.github.io.

*Corresponding author(s). E-mail(s): {shuxin.zheng, chang.liu, haiguang.liu, tie-yan.liu}@microsoft.com;
†These authors contributed equally to this work.

Abstract

Advances in deep learning have greatly improved structure prediction of molecules. However, many macroscopic observations that are important for real-world applications are not functions of a single molecular structure, but rather determined from the equilibrium distribution of structures. Traditional methods for obtaining these distributions, such as molecular dynamics simulation, are computationally expensive and often intractable. In this paper, we introduce a novel deep learning framework, called Distributional Graphormer (DiG), in an attempt to predict the equilibrium distribution of molecular systems. Inspired by the annealing process in thermodynamics, DiG employs deep neural networks to transform a simple distribution towards the equilibrium distribution, conditioned on a descriptor of a molecular system, such as a chemical graph or a protein sequence. This framework enables efficient generation of diverse conformations and provides estimations of state densities. We demonstrate the performance of DiG on several molecular tasks, including protein conformation sampling, ligand structure sampling, catalyst-adsorbate sampling, and property-guided structure generation. DiG presents a significant

advancement in methodology for statistically understanding molecular systems, opening up new research opportunities in molecular science.

Keywords: Equilibrium Distribution, Statistical Mechanics, Deep Learning, Molecular States

1 Main

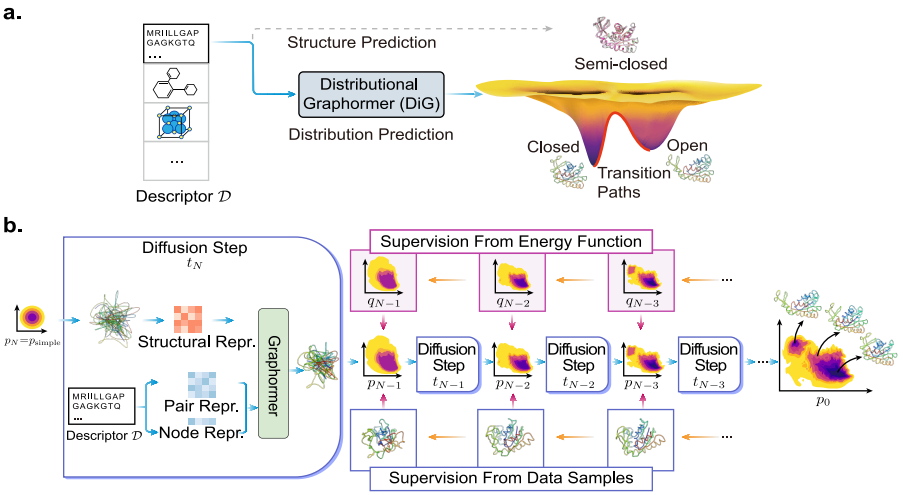

Fig. 1: **Predicting conformational distributions with the Distributional Graphormer (DiG) framework.** (a) DiG takes the basic descriptor $\mathcal{D}$ of a target molecular system as input, e.g., amino acid sequence, to generate a probability distribution of structures which aims at approximating the equilibrium distribution and sampling different metastable states or intermediate states. In contrast, static structure prediction methods, such as AlphaFold [1], aim at predicting one single high-probability structure of a molecule. (b) The DiG framework for predicting distributions of molecular structures. A deep-learning model (Graphormer [2]) is used as modules to predict a diffusion process ($\rightarrow$) that gradually transforms a simple distribution towards the target distribution. The model is learned so that the derived distribution p_i in each intermediate diffusion time step i matches the corresponding distribution q_i in a predefined diffusion process ($\leftarrow$) that is set to transform the equilibrium distribution to the simple distribution. Supervision can be obtained from both samples (lower row), and a molecular energy function (upper row).

Deep learning methods are now state of the art to predict structures of molecular systems with high efficiency. For example, AlphaFold achieves atomic-level accuracy in protein structure predictions [1], and has enabled new applications in structural biology [3–5]; fast docking methods based on deep neural networks have been developed and applied to predict ligand binding structures [6, 7], supporting virtual screening in drug discovery [8, 9]; deep learning models predict the relaxed structures of adsorbates on catalyst surfaces [2, 10–12]. All these developments demonstrate the potential of deep learning approaches in modeling molecular structures and states.

However, accurate prediction of the most probable structure only reveals a small portion of the information needed to understand a molecular system in equilibrium. In reality, molecules can be highly flexible and the equilibrium distribution is crucial for studying statistical mechanical properties. For example, functions of some biomolecules can be inferred from the probabilities associated with structures to identify metastable states; also based on probabilistic densities in the structure space, thermodynamic properties, such as entropy and free energies, can be computed by applying statistical mechanics methods.

Fig. 1a illustrates the difference between conventional structure prediction and the prediction of distributions of molecular structures. Although adenylate kinase has two distinct experimentally known conformations (open and closed states), a predicted structure usually corresponds to a highly probable metastable state or a low-probability intermediate state (as shown in this figure). A method is desired to allow us to sample the equilibrium distribution of adenylate kinase structures containing both functional states and their relative probabilities.

In contrast to the prediction of single structures, the prediction of equilibrium distributions still relies on classical and computationally expensive simulation methods while the development of deep learning methods for this task is still in its infancy. Most commonly, equilibrium distributions are sampled with molecule dynamics simulations which are computationally costly or even intractable [13]. Enhanced sampling simulations [14, 15] and Markov state modeling [16] can speed up rare event sampling, but rely on system-specific choices such as collective variables along which the sampling is enhanced, and is thus not an easily generalizable approach. A popular approach is coarse-grained molecular dynamics [17, 18] for which deep learning approaches have recently been developed [19, 20] that have shown promising results for individual molecular systems but not yet demonstrated generalization. Boltzmann Generators [21] are a deep learning approach to generate equilibrium distributions by constructing a probability flow from an easy-to-sample reference state, but due to the flow architecture [22] this approach is also difficult to generalize to different molecules. Generalization has been demonstrated for flows generating long timesteps for small peptides, but these methods have not yet scaled to large proteins [23].

In this work, we develop the Distributional Graphormer (DiG), a new deep learning approach aiming to approximately predict the equilibrium distribution and efficiently sample diverse and chemically plausible structures of molecular systems. We show that DiG can generalize across molecular systems and propose diverse structures for molecules not used during training that resemble experimentally known structures. DiG draws inspiration from simulated annealing [24–27], which produces a complex distribution by gradually refining a simple uniform distribution through the simulation of an annealing process. Following this idea, DiG reduces the difficulty in the equilibrium distribution prediction problem by simulating a diffusion process that gradually

transforms a simple distribution to the target distribution that aims at approximating the equilibrium distribution of the given molecular system [28, 29] (Fig. 1b, →). The diffusion process is realized by a deep-learning model that is based upon the Graphormer architecture (Fig. 1b, [2]), and that is conditioned on a descriptor of the target molecule, such as a chemical graph or an amino acid sequence. DiG can be trained using structure data from MD simulations and experiments. For cases where such data are not sufficient, we develop a novel Physics-Informed Diffusion Pre-training (PIDP) method to train DiG directly under the supervision from energy functions (force fields) of the systems. In both modes, the model receives a training signal in each diffusion step independently (Fig. 1b, ←), enabling efficient training that avoids backpropagating through the entire diffusion process.

The performance of DiG is evaluated on three prediction tasks: protein conformation distribution, ligand conformation distribution, and molecular adsorption distribution on catalyst surfaces. We demonstrate that DiG is capable of generating realistic and diverse molecular structures in these tasks. For the proteins shown in this paper, DiG efficiently generated structures to resemble major functional states, but with orders of magnitude less time than required for MD simulation. We also demonstrate that DiG can facilitate inverse design of molecular structures by applying biased distributions that favor structures with desired properties. This capability has the potential to broaden the scope of molecular design for properties that lack adequate data to guide the design process. These results indicate that DiG significantly advances deep learning methodology for molecules from predicting a single structure towards predicting probability distributions of molecular structures, paving the way for efficient prediction of thermodynamic properties of molecules.

2 The Framework of Distributional Graphormer

Deep neural networks have been demonstrated to predict accurate molecular structures from descriptors $\mathcal{D}$ for many molecular systems [1, 2, 6, 7, 10–12]. Here, DiG aims to take one step further to predict not only the most probable structure, but also diverse structures with probabilities under the equilibrium distribution. To tackle this challenge, inspired by the heating-annealing paradigm, we break down the difficulty of this problem into a series of simpler problems. The heating-annealing paradigm can be viewed as a pair of reciprocal stochastic processes on the structure space that simulate the transformation between the equilibrium distribution and a system-independent simple distribution p_{simple}. Following this idea, we employ an explicit diffusion process (forward process; Fig. 1b orange arrows) that gradually transforms the target distribution of the molecule $q_{\mathcal{D},0}$, as the initial distribution, towards p_{simple} through a time period τ. The corresponding reverse diffusion process then transforms p_{simple} back to the target distribution $q_{\mathcal{D},0}$. This is the generation process of DiG (Fig. 1b, blue arrows). The reverse process is performed by updates predicted by deep neural networks from the given $\mathcal{D}$, which are trained

to match the forward process. Compared to directly predicting the equilibrium distribution from $\mathcal{D}$, the heating-annealing paradigm significantly reduces the difficulty of this problem. As p_{simple} is chosen to enable independent sampling and have a closed-form density function, DiG enables independent sampling of the equilibrium distribution by simulating the reverse process started from p_{simple}, and also provides a density function for the distribution by tracking the process.

Specifically, we choose $p_{\text{simple}} := \mathcal{N}(\mathbf{0}, \mathbf{I})$ as the standard Gaussian distribution in the state space, and the forward diffusion process as the Langevin diffusion process targeting this p_{simple} (Ornstein–Uhlenbeck process) [30–32]. A time dilation scheme β_t [33] is introduced for approximate convergence to p_{simple} after a finite time τ. The result is written as the following stochastic differential equation (SDE):

$$\mathrm{d}\mathbf{R}_t = -\frac{\beta_t}{2}\mathbf{R}_t\,\mathrm{d}t + \sqrt{\beta_t}\,\mathrm{d}\mathbf{B}_t, \tag{1}$$

where $\mathbf{B}_t$ is the standard Brownian motion (a.k.a Wiener process). Choosing this forward process leads to a p_{simple} that is more concentrated than a heated distribution hence it is easier to draw high-density samples, and the form of the process enables efficient training and sampling.

Following stochastic process theory (e.g., [34]), the reverse process is also a stochastic process, written as the following SDE:

$$\mathrm{d}\mathbf{R}_{\bar{t}} = \frac{\beta_{\bar{t}}}{2}\mathbf{R}_{\bar{t}}\,\mathrm{d}\bar{t} + \beta_{\bar{t}}\nabla \log q_{\mathcal{D},\bar{t}}(\mathbf{R}_{\bar{t}})\,\mathrm{d}\bar{t} + \sqrt{\beta_{\bar{t}}}\,\mathrm{d}\mathbf{B}_{\bar{t}}, \tag{2}$$

where $\bar{t} := \tau - t$ is the reversed time, $q_{\mathcal{D},\bar{t}} := q_{\mathcal{D},t=\tau-\bar{t}}$ is the forward-process distribution at the corresponding time, and $\mathbf{B}_{\bar{t}}$ is the Brownian motion in reversed time. To recover $q_{\mathcal{D},0}$ from p_{simple} by simulating this reverse process, deep neural networks are employed to construct a score model $\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R})$, which is trained to predict the true score function $\nabla \log q_{\mathcal{D},t}(\mathbf{R})$ of each instantaneous distribution $q_{\mathcal{D},t}$ from the forward process. This formulation is called diffusion-based generative model and has been demonstrated to be able to generate high-quality samples of images and other content [28, 29, 35–37]. As our score model is defined in molecular conformational space, we employ our previously developed Graphormer model [2] as the neural network architecture backbone of DiG, to leverage its capabilities in modeling molecular structures and to generalize to a range of molecular systems.

With the $\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R})$ model, drawing a sample $\mathbf{R}_0$ from the equilibrium distribution of a system $\mathcal{D}$ can be done by simulating the reverse process Eq. (2) on $N+1$ steps that uniformly discretizes $[0,\tau]$ with step size $h = \tau/N$ (Fig. 1b,

blue arrows):

$$\begin{aligned}&\mathbf{R}_N \sim p_{\text{simple}},\\&\mathbf{R}_{i-1} = \frac{1}{\sqrt{1-\beta_i}}\Big(\mathbf{R}_i + \beta_i \mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i)\Big) + \mathcal{N}(\mathbf{0}, \beta_i \mathbf{I}),\ i = N, \cdots, 1,\end{aligned} \tag{3}$$

where the discrete step index i corresponds to time $t = ih$, and $\beta_i := h\beta_{t=ih}$. Note that the reverse process does not need to be ergodic. The way that DiG models the equilibrium distribution is using the instantaneous distribution at the instant $t = 0$ (or $\bar{t} = \tau$) on the reverse process, but not using a time average. As $\mathbf{R}_N$ samples can be drawn independently, DiG can generate statistically independent $\mathbf{R}_0$ samples for the equilibrium distribution. In contrast to Molecular Dynamics (MD) or Markov Chain Monte Carlo (MCMC) simulations, generation of DiG samples does not suffer from rare events, and can thus be far more computationally efficient.

Physics-Informed Diffusion Pre-training

DiG can be trained by conformation data sampled over a range of molecular systems. However, collecting sufficient experimental or simulation data to characterize the equilibrium distribution for various systems is extremely costly. To address this data scarcity problem, we propose a novel pre-training algorithm, called Physics-Informed Diffusion Pre-training (PIDP), which effectively optimizes DiG on an initial set of candidate structures that need not to be sampled from the equilibrium distribution. The supervision comes from the energy function $E_{\mathcal{D}}$ of each system $\mathcal{D}$, which defines the equilibrium distribution $q_{\mathcal{D},0}(\mathbf{R}) \propto \exp(-\frac{E_{\mathcal{D}}(\mathbf{R})}{k_{\mathrm{B}}T})$ at the target temperature T.

The key idea is that the true score function $\nabla \log q_{\mathcal{D},t}$ from the forward process Eq. (1) obeys a partial differential equation, known as the Fokker-Planck equation (e.g., [38]). We then pre-train the score model $\mathbf{s}^{\theta}_{\mathcal{D},t}$ by minimizing the following loss function that enforces the equation to hold:

$$\begin{aligned}&\sum_{i=1}^{N}\frac{1}{M}\sum_{m=1}^{M}\Big\|\frac{\beta_i}{2}\Big(\nabla\big(\mathbf{R}^{(m)}_{\mathcal{D},i}\cdot\mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}^{(m)}_{\mathcal{D},i})\big) + \nabla\|\mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}^{(m)}_{\mathcal{D},i})\|^2 + \nabla\big(\nabla\cdot\mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}^{(m)}_{\mathcal{D},i})\big)\Big)\\&\quad - \frac{\partial}{\partial t}\mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}^{(m)}_{\mathcal{D},i})\Big\|^2 + \frac{\lambda_1}{M}\sum_{m=1}^{M}\Big\|\frac{1}{k_{\mathrm{B}}T}\nabla E_{\mathcal{D}}(\mathbf{R}^{(m)}_{\mathcal{D},1}) + \mathbf{s}^{\theta}_{\mathcal{D},1}(\mathbf{R}^{(m)}_{\mathcal{D},1})\Big\|^2.\end{aligned} \tag{4}$$

Here, the second term, weighted by λ_1, matches the score model at the final generation step to the score from the energy function, and the first term implicitly propagates the energy-function supervision to intermediate time steps (Fig. 1b, upper row). The structures $\{\mathbf{R}^{(m)}_{\mathcal{D},i}\}_{m=1}^{M}$ to evaluate the loss are points on a grid spanning the structure space. What is favorable is that, these structures do not have to obey the equilibrium distribution (as is required by data structures), since they are only used to discretize functions in the structure space, therefore the cost of preparing these structures can be much lower. As structure spaces of molecular systems are often very high-dimensional (e.g.,

thousands for proteins), a regular grid would have intractably many points. Fortunately, the space of actual interest is only a low-dimensional manifold of physically reasonable structures (structures with low energy) relevant to the problem. This allows us to effectively train the model only on these relevant structures as $\mathbf{R}_0$ samples, and pass them through the forward process for $\mathbf{R}_i$ samples. See Supplementary Sec. C.1 for an example on acquiring relevant structures for protein systems.

We also leverage stochastic estimators including Hutchinson's estimator [39, 40] to reduce the complexity in calculating derivatives of high-order and for high-dimensional vector-valued functions. Note that for each step i, the corresponding model $\mathbf{s}^{\theta}_{\mathcal{D},i}$ receives a training loss independent of other steps and can be directly back-propagated. This step-by-step supervision pattern helps to achieve efficient pre-training.

Training DiG with Data

In addition to using the energy function for information on the probability distribution of the molecular system, DiG can also be trained with molecular structure samples which can be obtained from experimental structure determination methods, molecular dynamics, or other simulation methods. See Supplementary Sec. C for data collection details. Even when the simulation data is limited, they still provide information about the regions the distribution needs to cover and the local shape of the distribution, hence are helpful to improve a pre-trained DiG. To train DiG on data, the score model $\mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i)$ is matched to the corresponding score function $\nabla \log q_{\mathcal{D},i}$ demonstrated by data samples. This can be done by minimizing $\mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \left\| \mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i) - \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2$ for each diffusion time step i. Although the precise calculation of $\nabla \log q_{\mathcal{D},i}$ is impractical, the loss function can be equivalently reformulated into denoising score-matching form [41, 42]:

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)} \mathbb{E}_{p(\epsilon_i)} \left\| \sigma_i \mathbf{s}^{\theta}_{\mathcal{D},i}(\alpha_i \mathbf{R}_0 + \sigma_i \epsilon_i) + \epsilon_i \right\|^2, \tag{5}$$

where $\alpha_i := \prod_{j=1}^{i} \sqrt{1-\beta_j}$, $\sigma_i := \sqrt{1-\alpha_i^2}$, and $p(\epsilon_i)$ is the standard Gaussian distribution. The expectation under $q_{\mathcal{D},0}$ can be estimated using the simulation dataset. Note that this function allows direct loss estimation and backpropagation for each i in constant (w.r.t i) cost, recovering the efficient step-by-step supervision again (Fig. 1b, lower row).

Density Estimation by DiG

Many thermodynamic properties of a molecular system (e.g., free energy, entropy) also require calculating the density function of the equilibrium distribution, which is another aspect of the distribution besides a sampling method. DiG allows for this by tracking the distribution change along the diffusion

process [35]:

$$\log p^{\theta}_{\mathcal{D},0}(\mathbf{R}_0) = \log p_{\text{simple}}\big(\mathbf{R}^{\theta}_{\mathcal{D},\tau}(\mathbf{R}_0)\big) - \int_0^{\tau} \frac{\beta_t}{2} \nabla \cdot \mathbf{s}^{\theta}_{\mathcal{D},t}\big(\mathbf{R}^{\theta}_{\mathcal{D},t}(\mathbf{R}_0)\big)\, \mathrm{d}t - \frac{D}{2} \int_0^{\tau} \beta_t \, \mathrm{d}t, \tag{6}$$

where D is the dimension of the state space, and $\mathbf{R}^{\theta}_{\mathcal{D},t}(\mathbf{R}_0)$ is the solution to the ordinary differential equation (ODE):

$$\mathrm{d}\mathbf{R}_t = -\frac{\beta_t}{2}\Big(\mathbf{R}_t + \mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\Big)\, \mathrm{d}t, \tag{7}$$

with initial condition $\mathbf{R}_0$, which can be solved using standard black box ODE solvers or more efficient specific solvers (Supplementary Sec. A.6).

Property-Guided Structure Generation with DiG

There is a growing demand for inverse design of materials and molecules. The goal is to find structures with desired properties, such as intrinsic electronic band gaps, elastic modulus, and ionic conductivity, without going through a forward searching process. DiG provides a feature to enable such property-guided structure generation, by directly predicting the conditional structural distribution given a value c of a microscopic property.

To achieve this, regarding the data-generating process in Eq. (2), we only need to adapt the score function, from $\nabla \log q_{\mathcal{D},t}(\mathbf{R})$ to $\nabla_{\mathbf{R}} \log q_{\mathcal{D},t}(\mathbf{R} \mid c)$. Using Bayes' rule, the latter can be reformulated as $\nabla_{\mathbf{R}} \log q_{\mathcal{D},t}(\mathbf{R} \mid c) = \nabla \log q_{\mathcal{D},t}(\mathbf{R}) + \nabla_{\mathbf{R}} \log q_{\mathcal{D}}(c \mid \mathbf{R})$, where the first term can be approximated by the learned (unconditioned) score model, i.e. the new score model is:

$$\mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i \mid c) = \mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i) + \nabla_{\mathbf{R}_i} \log q_{\mathcal{D}}(c \mid \mathbf{R}_i). \tag{8}$$

Hence, only a $q_{\mathcal{D}}(c \mid \mathbf{R})$ model is additionally needed [35, 36], which is a property predictor or classifier that is much easier to train than a generative model.

It is noted that in a normal workflow for machine-learning (ML) inverse design, a dataset must be generated to meet the conditional distribution, then an ML model will be trained on this dataset for structure predictions. The ability to generate structures for conditional distribution without requiring a conditional dataset places DiG in an advantageous position when compared to the normal workflow in terms of efficiency and computational cost.

Interpolation between States

Given two states, DiG can approximate a reaction path that corresponds to reaction coordinates or collective variables, and find intermediate states along the transition pathway. This is achieved through the fact that the distribution transformation process described in Eq. (1) is equivalent to the process in

Eq. (7) if $\mathbf{s}^{\theta}_{\mathcal{D},i}$ is well learned, which is deterministic and invertible hence establishes a correspondence between the structure and latent space. We can then uniquely map the two given states in the structure space to the latent space, approximate the path in the latent space by linear interpolation, and then map the path back to the structure space. Since the distribution in the latent space is Gaussian which has a convex contour, the linearly interpolated path goes through high-probability or low-energy regions, so it gives an intuitive guess of the real reaction path.

3 Results

Here, we demonstrate that DiG can be applied to study protein conformations, protein-ligand interactions, and molecule adsorption on catalysis surfaces. In addition, we investigate the inverse design capability of DiG, through its application to carbon polymorph generation for desired electronic band gaps.

3.1 Protein Conformation Sampling

At physiological conditions, most protein molecules exhibit dynamical behaviors, rather than existing as rigid objects in their most energetically favorable states. The sampling of these conformations is crucial for the comprehensive understanding of protein properties and their interactions with other molecules in cells. Recently, it has been reported that AlphaFold [1] can generate alternative conformations for certain proteins, by manipulating input information such as multiple sequence alignments (MSA) [43]. However, this approach is developed on the basis of varying the depth of MSA, it is hard to generalize to all proteins (especially for those with a small number of similar sequences). Therefore, it is highly desirable to have advanced AI models that can sample diverse structures consistent with the energy landscape in the conformational space [43]. Here, we show that DiG is capable of generating diverse and functionally relevant protein structures, which is a key capability for being able to efficiently sample equilibrium distributions.

It is noted that the equilibrium distribution of protein conformations is difficult to obtain experimentally or computationally, so in contrast to protein structure prediction, there is a lack of high-quality data for training or benchmarking. To train this model, we collect experimental and simulated structures from public databases. In order to mitigate the data scarcity issue, besides the structures from the protein databank, we also generated an in-house simulation dataset and developed the PIDP training method (See Supplementary Sec. A.1.1 and D.1 for training procedure and the dataset). The performance of DiG was assessed at two levels: (1) comparing the conformational distributions against those obtained from extensive (millisecond timescale) atomistic MD simulations; (2) validating on proteins with multiple known conformations. As shown in Fig. 2a, the conformational distributions are obtained from MD simulations for two proteins from the SARS-CoV-2 virus [44] (the receptor-binding-domain (RBD) of spike protein and the main protease, also known as

3CL protease, see Supplementary Sec. A.7 for details on MD simulation data). These two proteins are the crucial components of the SARS-CoV-2 virus and key targets for drug development in the treatment of COVID-19 [45, 46]. The millisecond timescale MD simulations extensively sample conformation space, and we therefore regard the resulting distribution as a proxy to the equilibrium distribution. Taking protein sequences as the descriptor inputs for DiG, structures were generated for these two proteins. Although MD simulation

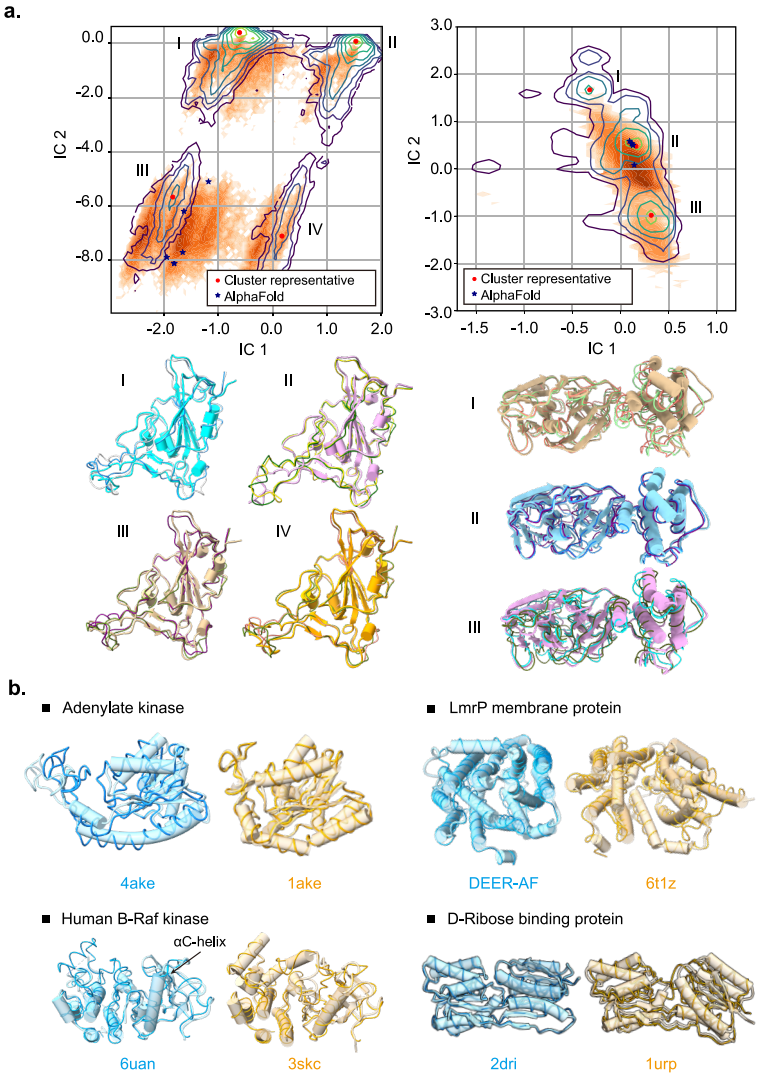

Fig. 2: **Distribution and sampling results for protein conformations.**

Fig. 2: (a) Structures generated by DiG resemble the diverse conformations of millisecond MD simulations. MD simulated structures are projected onto the reduced 2D space spanned by TICA coordinates, and the probability densities are depicted using contour lines. For RBD protein, MD simulation reveals four highly populated regions in the 2D space spanned by TICA coordinates (left panel). Structures generated by DiG are mapped to this 2D space shown as orange dots, whose distributions are reflected by the color intensity. Below the distribution map, structures generated by DiG (thin ribbons) are superposed to representative structures of four clusters. AlphaFold predicted structures ($\star$) are also shown in the plot. Right panel shows the results of the main protease of SARS-CoV-2, compared with MD simulations and AlphaFold prediction results. The contour map reveals three clusters, DiG generates highly similar structures in cluster II & III, while structures in cluster-I are accurately generated. (b) The performance of DiG on generating multiple conformations of proteins (each structure is labeled by its PDB ID, except the DEER-AF, which is AlphaFold predicted model that is consistent with experimental observations). Structures generated by DiG (thin ribbons) are compared with the experimentally determined structures (cylindrical cartoons) in each case. For the four proteins (adenylate kinase, Lmrb membrane protein, human B-Raf kinase, and D-ribose binding protein), structures in two functional states (distinguished by cyan and brown) are well reproduced by DiG (ribbons).

data of these proteins were not used for DiG training, the generated structures resemble the conformational distributions explored by MD in the reduced dimension space spanned by collective variables (Fig. 2a). In the 2D projection shown here, the MD simulations of RBD populate four regions, which are also sampled by DiG (see Fig. 2a, left panel). The four representative structures corresponding to the cluster centers are well generated by DiG. Similarly, three representative structures for main protease were obtained by clustering analysis on MD simulation trajectories, and then the generated structures were aligned to these three representatives (Fig. 2a). We noticed that conformations in cluster-I region are not well recovered by DiG, indicating room for improvement. In terms of conformational space coverage, we compared the DiG sampled regions with those explored by MD simulations in the conformation manifold spanned by the TICA variables (Fig. 2a). For example, on the 2D manifold, about 70% of the RBD conformations sampled by millisecond-scale MD simulations can be covered with just 10,000 DiG-generated structures (see Supplementary Fig. S1 for details).

Atomistic MD simulations are computationally very expensive, therefore millisecond time scale simulations of proteins are rarely reported in literature, except for simulations on special-purpose hardware such as the Anton supercomputer [13] or extensive distributed simulations combined in Markov state models [16]. In order to get an additional assessment on the diversity of protein structures generated by DiG, we turn to proteins for which multiple

structures have been experimentally determined. Although it is a less stringent test, the capability of sampling alternative conformations can facilitate the research of protein dynamics and functional mechanisms. We analyzed four proteins, each with two distinguishable conformations corresponding to different functional states (Fig. 2b). Remarkably, the conformations sampled by DiG have good coverage in the conformational space near the two states for each protein. The experimentally determined conformations are shown in cylinder cartoons, each aligned with two structures generated by DiG (shown in ribbon representations). For example, the adenylate kinase protein has two conformations (PDB IDs 1ake and 4ake), each with high-quality structures in their vicinity (backbone RMSD < 1.0 Å for the structure superposed to the closed state, 1ake; backbone RMSD < 3.0 Å for the structures superposed to the open state, 4ake). Similarly, for the drug transport protein LmrP, DiG generated structures resembling both states. We note that one structure is experimentally determined, and the other (denoted as DEER-AF) is the AlphaFold predicted structure [43] supported by double electron electron resonance (DEER) experimental data [47]. For the case of human B-Raf kinase, the overall RMSD difference between the two experimentally determined states is not as pronounced as in the other three proteins. The major structural difference is in the A-loop region and a nearby helix (α C-helix, indicated in the figure) [48]. Structures generated by DiG accurately recover such regional structural differences in this kinase protein. Another interesting case is the D-Ribose binding protein with two separated domains, which can be packed in two distinct conformations. DiG correctly generates structures corresponding to both the straight-up conformation (cylinder cartoon) and the twisted/tilted conformation. It is noted that if we align one domain of D-ribose binding protein, the other domain only partially matches the twisted conformation as an 'intermediate' state. Furthermore, for a pair of structures of the same protein, DiG can be applied to generate transition pathways by latent space interpolations (see demonstration cases in the DiG webpage: https://DistributionalGraphormer.github.io). The dynamics revealed by such pathways can inspire hypotheses on molecular mechanisms for experimental validation. In summary, DiG is capable of generating diverse protein structures corresponding to different functional states, thus going beyond the capabilities of current static structure prediction methods.

3.2 Ligand Structure Sampling around Binding Sites

An immediate extension of protein conformational sampling is to predict protein-ligand interactions, such as ligand binding positions in druggable pockets. To model the interactions between protein and ligand, we mainly use a simulation dataset of about 1500 complexes for training (See Supplementary Sec. D.1 for the dataset). We evaluated the performance of DiG in ligand binding to protein pockets for 409 protein-ligand systems [49, 50] (not in the training dataset). By providing atomic positions surrounding a pocket and a ligand descriptor (here, a SMILES string), DiG generates ligand structures to

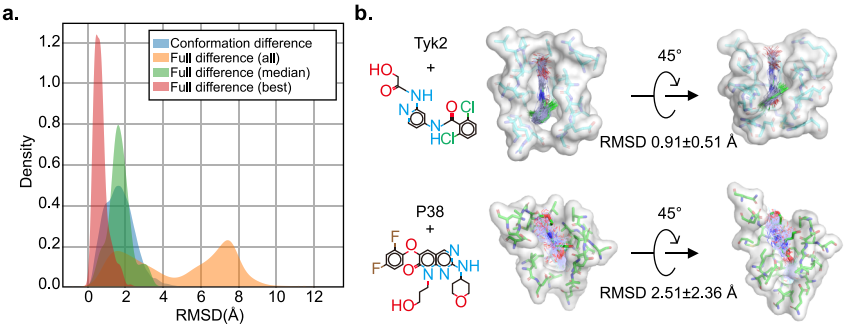

Fig. 3: **Results of DiG for ligand structure sampling around protein pockets.** (a) The results of DiG on poses of ligands bound to protein pockets. DiG generates ligand structures and binding poses, with good accuracy compared to the crystal structures (reflected by the RMSD statistics shown in red histogram for the best matching cases, and the green histogram for the median RMSD statistics). When considering all 50 predicted binding poses for each system, diversity is observed, as reflected in the RMSD histogram (yellow color, normalized) compared to the references. (b) Representative systems show that the diversity in ligand binding poses is related to the binding pocket properties. For deep and narrow binding pocket such as for the Tyk2 protein (shown in the surface representation, top panel), DiG predicts highly similar binding poses for the ligand (in atom-bond representations, top panel). For the P38 protein the binding pocket is relatively flat and shallow and predicted ligand poses are highly diverse and have large conformational flexibility (bottom panel, in the same representations as in the Tyk2 case).

fit the pocket. During the ligand structure sampling, DiG models the atomic coordinate distribution of both binding pocket and the ligand. The flexible binding pockets were observed in the testing, with changes in atomic positions up to 1.0 Å in terms of RMSD compared to the input atomic positions. For the ligand structures, the deviation comes from two sources: (1) the conformational difference between generated structures and experimental structures; and (2) the difference in the binding pose due to ligand translation and rotation. Among all tested cases, the conformational differences are small, with an RMSD value of 1.74 Å on average, indicating that generated ligand structures are highly similar to the bound ligands resolved in crystal structures (Fig. 3a). When including the binding pose deviations originated from ligand positions and orientations, larger alignment discrepancies are observed for ligand structures. Yet, the DiG is still capable of predicting at least one correct structure for each ligand out of 50 generated structures. In a retrospective measurement, the best-matched structure among 50 generated structures for each ligand is within 2.0 Å RMSD compared to the experimental data for nearly all 409 testing systems (Fig. 3a for the RMSD distribution, with more cases shown in Supplementary Fig. S3). The accuracy of generated structures for ligand is

related to the characteristic of binding pockets. For example, the ligand binding to the target protein Tyk2 showed an average deviation of 0.91 Å (RMSD) from the crystal structure (see Fig. 3b, top). In another example for target P38, the ligand exhibited more diverse binding poses, likely due to the shallow pocket of this target. Under such circumstances, the most stable binding pose may be less dominant compared to other favorable poses (Fig. 3b, bottom). MD simulations reveal similar trends as DiG-generated structures, with ligand binding to Tyk2 more tightly than the case of P38 (Supplementary Fig. S2). Overall, we observed that the generated structures indeed resemble experimentally observed poses.

3.3 Catalyst-Adsorbate Sampling

Identifying active adsorption sites is a central task in heterogeneous catalysis. Due to complex surface-molecular interactions, such tasks rely heavily on a combination of quantum chemistry methods such as density functional theory (DFT) and sampling techniques such as MD and grid-search. These lead to large and sometimes intractable computational costs, especially when it comes to surfaces with complex chemical environments. We evaluate DiG's capability for this task by training it on the MD trajectories of catalyst-adsorbate systems from the Open Catalyst Project and carrying out further evaluations on random combinations of adsorbates and surfaces that are not included in the training set [10]. By feeding the model with a substrate and a molecular adsorbate, DiG can predict adsorption sites and stable adsorbate configurations, along with the probability for each configuration (see Supplementary Sec. A.4 for training details and Supplementary Sec. A.7 for evaluation details). Fig. 4a-b shows the adsorption configurations of an acyl group on a stepped TiIr alloy surface. Multiple adsorption sites are predicted by DiG. To test the plausibility of these predicted configurations and evaluate the coverage of the predictions, we carry out a grid-search using DFT methods. The results confirm that DiG predicts all stable sites found by the grid-search and the adsorption configurations are in close agreement with an RMSD of $0.5 \sim 0.8$ Å (Fig. 4b). It should be noted that the combination of substrate and adsorbate shown in Fig. 4b is not included in the training data set. Therefore, the result demonstrates the cross-system generalization capability of DiG in catalyst adsorption predictions. Here we show only the top view, and Fig. S4 in addition shows the front view of the adsorption configurations.

DiG not only predicts the adsorption sites with correct configurations, but also provides a probability estimate for each adsorption configuration. This capability is illustrated in the systems with single-atom adsorbates (including H, N, and O) on 10 randomly chosen metallic surfaces. For each combination of adsorbate and catalyst substrate, the DiG is applied to predict the adsorption sites and the probability distributions. Then for the same systems, grid-search DFT calculations were carried out to find all adsorption sites and the corresponding energies. Taking the adsorption sites identified by grid-search as references, DiG achieved 81% site coverage for single-atom adsorbates on the

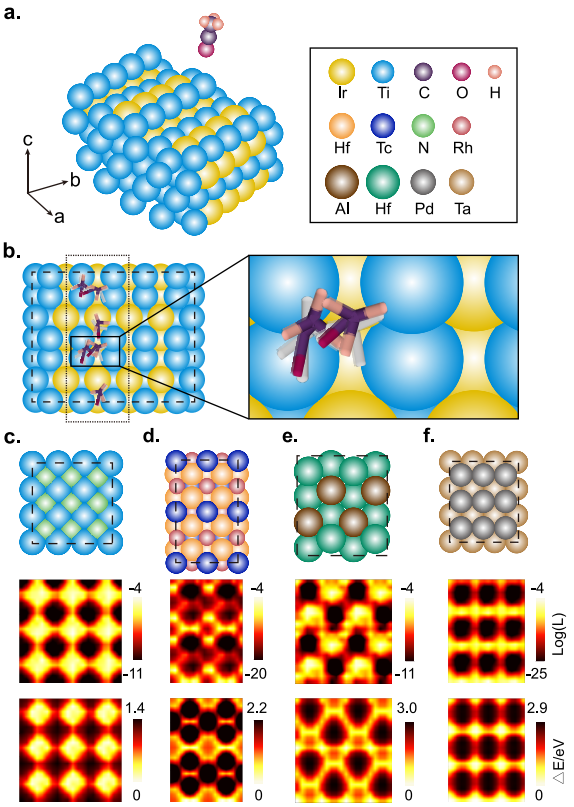

Fig. 4: **Results of DiG for catalyst-adsorbate sampling problems.** (a) The problem setting: prediction of the adsorption configuration distribution of an adsorbate on a catalyst surface. (b) The adsorption sites and corresponding configurations of the adsorbate found by DiG (in color), compared with DFT results (in white). DiG finds all the adsorption sites, with adsorbate structures close to the DFT baseline. For all adsoprtion sites and configurations, refer to Appendix E. (c-f) Adsorption prediction results of single N and O atoms on catalyst surfaces, compared to DFT calculations. Top panels show the catalyst surface; the probability distribution of adsorbate molecules on the corresponding catalyst surfaces are shown in the middle panels in log-scale; the bottom panels show the calculated interactions between the adsorbate molecule and the catalyst using DFT methods. The adsorption sites and predicted probabilities are highly consistent with the energy landscape obtained by DFT computations.

10 metallic catalyst surfaces. Fig. 4(c-f) show closer examinations on adsorption predictions for four systems, namely C, H, N, and O on TiN, RhTcHf, AlHf, and TaPd metallic surfaces (top panels). The predicted adsorption probabilities projected on the surface in parallel with the catalyst surface are shown

in the middle panels. The log-scaled heatmaps of the probabilities show excellent accordance with the adsorption energies calculated using DFT methods (bottom panels). It is worth noting that the speed of DiG is much faster compared to DFT, i.e., it only takes about 1 minute to sample all adsorption sites for a catalyst-adsorbate system for DiG on a single modern GPU, but at least 2 hours for a single DFT relaxation with VASP, which number will be further multiplied by a factor of > 100 depending on the resolution of the searching grid [51]. Such fast and accurate prediction of adsorption sites and the corresponding distributional features can be useful in identifying the catalytic mechanisms and guiding the search of new catalysts.

3.4 Property-Guided Structure Generation

While DiG by default generates structures following the learned training data distribution, the output distribution can be biased to steer the structure generation to meet particular requirements. Here we leverage this capability by employing DiG for inverse design (described in Sec. 2). As a proof-of-concept, we search for carbon polymorphs with desired electronic band gaps. Similar tasks are critical to the discovery of novel photovoltaic and semi-conductive materials [52]. To train this model, we prepared a structure dataset composed of carbon atoms by carrying out random structure search based on energy profiles obtained from DFT calculations [53]. The structures corresponding to energy minima form the dataset used to train DiG, which in turn are applied to generate carbon structures. We use a neural network model based on the M3GNet architecture [11] as the property predictor for band gap, which is fed to the property-guided structure generation of carbon structures.

Fig. 5 shows the distributions of band gaps calculated from generated carbon structures. In the original training dataset, most structures have a band gap around 0 eV (see Fig. 5a). When the target band gaps are supplied to DiG, the structures are generated with the desired band gaps. With the guidance of a band gap model in conditional generation, the distribution is biased towards the targets, showing pronounced peaks around the target band gaps. Representative structures are shown in Fig. 5. For conditional generation with a target band gap of 4 eV, DiG generates stable carbon structures similar to diamond, which has large band gaps. In the case of 0 eV band gap, we obtain graphite-like structures with low band gaps. In Fig. 5a, we show some structures by unconditional generation. To evaluate the quality of carbon crystal structures generated by DiG, we calculate the ratio of structures that match one of the relaxed structures in the dataset by using the `StructureMatcher` in the PyMatgen package [54]. For unconditional generation, the match rate is 99.87%, and the average matched normalized RMSD computed from fractional coordinates over all sampled structures is 0.16. For conditional generation, the match rate is 99.99%, but with a higher average normalized RMSD of 0.22. While increasing the possibility of generating structures with target band gap, conditional generation can influence the quality of the structures (see Supplementary Sec. F.1 for more discussions). This proof-of-concept study shows

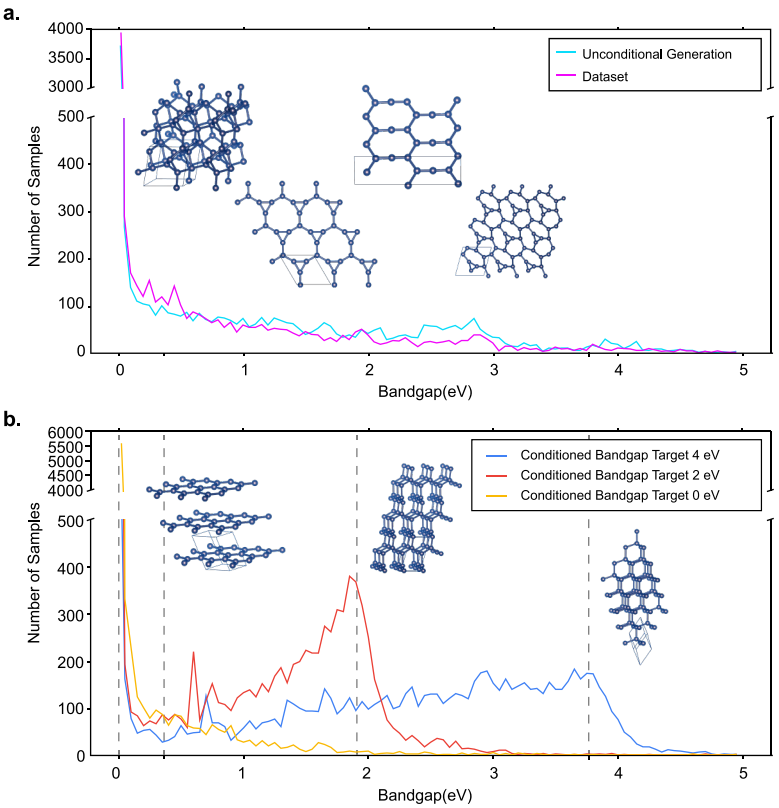

Fig. 5: **Property-guided structure generation of carbon structures with particular band gaps.** (a) Electronic band gaps of generated structures from trained DiG with no specification on the desired band gap. The generated structures do not show any obvious preference on band gaps, closely resembling the distribution of the training dataset. (b) Structure generated for three band gaps (0, 2, and 4 eV). The distributions of band gaps for generated structures peak at the desired values. In particular, DiG generates graphite-like structures when desired band gap is 0 eV; while at 4 eV band gap, the generated structures are most similar to diamonds. Representative structures are shown in the inset of the plot.

that DiG not only captures the probability distributions with complex features in large configurational space, but also can be applied for inverse materials design, when combined with a property quantifier, such as an ML predictor. Since the training of the property prediction model (e.g., the M3GNet band gap model) and the diffusion model of DiG are fully decoupled, our approach can be readily extended to inverse design for other properties.

4 Discussion

Predicting the equilibrium distribution of molecular states is a formidable challenge in the molecular sciences, with far-reaching implications for deciphering structure-function relationships, computing macroscopic properties, and designing novel molecules and materials. With existing methods, a vast number of measurements or simulated samples of single molecules are required to gather sufficient data for characterizing the equilibrium distribution. We introduce Distributional Graphormer (DiG), a deep generative framework capable of predicting probability distributions which enables efficiently sampling diverse conformations and estimating their state densities across molecular systems. Drawing inspiration from the annealing process, DiG employs a sequence of deep neural networks to progressively transform state distributions from a simplistic mathematical form to the target distributions which can be trained to approximate the equilibrium distribution with suitable training data.

We have applied DiG to several molecular prediction tasks, including protein conformation sampling, protein-ligand binding structure generation, molecular adsorption on catalyst surfaces, and property-guided structure generation. The results show that DiG is capable of generating chemically realistic and diverse structures, and distributions resembling those of extensive MD simulations in low dimensional projections in some cases. By harnessing the power of advanced deep learning architectures, DiG can learn the representation of molecular conformations that are transformed from molecular descriptors, such as amino acid sequences for proteins or chemical formulas for compound molecules. Furthermore, its capacity to model complex, multimodal distributions using diffusion models enables it to capture equilibrium distributions in high-dimensional space.

DiG has been demonstrated to be capable to generalize across molecules within the same class, such as in the case of proteins, small molecules, and catalyst structures. Consequently, the framework opens the door to a multitude of research opportunities and applications in molecular science. Thus, when fed with suitably distributed training data, DiG can provide insights into the statistical understanding of molecules, enabling the computing of macroscopic properties, such as free energies and thermodynamic stability. These insights are critical for investigating the physical and chemical phenomena of molecular systems.

Finally, with its capability in generating independent and identically distributed (i.i.d.) conformations from equilibrium distributions, DiG offers a significant computational advantage over traditional sampling or simulation approaches that suffer from rare events, such as MCMC or MD simulations. DiG achieves similar conformation space coverage as millisecond-timescale MD simulations do in the two tested protein cases. Based on the OpenMM benchmark performance of modern GPU devices, it would require about 7-10 GPU years on Nvidia A100s to complete a simulation of 1.8 ms for RBD of the spike protein; while generating 50k structures using DiG only takes about 10 days on

a single A100 GPU without any inference acceleration (see more discussion in Supplementary Sec. A.6). Similar levels of speedup can be achieved in the case of predicting adsorbate distribution on the catalyst surface, as elaborated in the result section. If such order-of-magnitude speed-up can be combined with generating high-accuracy probability distributions, this will be transformative for molecular simulation and design.

While the quantitative prediction of equilibrium distributions at given thermodynamic states will hinge upon the availability of training data, the capacity of DiG to explore vast and diverse conformational spaces contributes to the discovery of novel and functional molecular structures, including protein structures, ligand conformers, and adsorbate configurations. DiG can therefore help to bridge the gap between microscopic descriptors and macroscopic observations of molecular systems, with potential impact on various areas of molecular sciences including life sciences, drug design, catalysis research, and materials sciences.

References

[1] 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)

[2] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.-Y.: Do transformers really perform badly for graph representation? Advances in Neural Information Processing Systems **34**, 28877–28888 (2021)

[3] Cramer, P.: Alphafold2 and the future of structural biology. Nature structural & molecular biology **28**(9), 704–705 (2021)

[4] Akdel, M., Pires, D.E., Pardo, E.P., Jänes, J., Zalevsky, A.O., Mészáros, B., Bryant, P., Good, L.L., Laskowski, R.A., Pozzati, G., et al.: A structural biology community assessment of alphafold2 applications. Nature Structural & Molecular Biology, 1–12 (2022)

[5] Pereira, J., Simpkin, A.J., Hartmann, M.D., Rigden, D.J., Keegan, R.M., Lupas, A.N.: High-accuracy protein structure prediction in casp14. Proteins: Structure, Function, and Bioinformatics **89**(12), 1687–1699 (2021)

[6] Stärk, H., Ganea, O., Pattanaik, L., Barzilay, R., Jaakkola, T.: Equibind: Geometric deep learning for drug binding structure prediction. In: International Conference on Machine Learning, pp. 20503–20521 (2022). PMLR

[7] Corso, G., Stärk, H., Jing, B., Barzilay, R., Jaakkola, T.: DiffDock: Diffusion steps, twists, and turns for molecular docking. In: International Conference on Learning Representations (2023)

[8] Diaz-Rovira, A.M., Martin, H., Beuming, T., Diaz, L., Guallar, V., Ray, S.S.: Are deep learning structural models sufficiently accurate for virtual screening? application of docking algorithms to alphafold2 predicted structures. bioRxiv, 2022–08 (2022)

[9] Scardino, V., Di Filippo, J.I., Cavasotto, C.N.: How good are alphafold models for docking-based virtual screening? Iscience **26**(1) (2023)

[10] Chanussot, L., Das, A., Goyal, S., Lavril, T., Shuaibi, M., Riviere, M., Tran, K., Heras-Domingo, J., Ho, C., Hu, W., *et al.*: Open catalyst 2020 (oc20) dataset and community challenges. ACS Catalysis **11**(10), 6059–6072 (2021)

[11] Chen, C., Ong, S.P.: A universal graph deep learning interatomic potential for the periodic table. Nature Computational Science **2**(11), 718–728 (2022)

[12] Schaarschmidt, M., Riviere, M., Ganose, A.M., Spencer, J.S., Gaunt, A.L., Kirkpatrick, J., Axelrod, S., Battaglia, P.W., Godwin, J.: Learned force fields are ready for ground state catalyst discovery. arXiv preprint arXiv:2209.12466 (2022)

[13] Lindorff-Larsen, K., Piana, S., Dror, R.O., Shaw, D.E.: How fast-folding proteins fold. Science **334**(6055), 517–520 (2011)

[14] Barducci, A., Bonomi, M., Parrinello, M.: Metadynamics. Wiley Interdisciplinary Reviews: Computational Molecular Science **1**(5), 826–843 (2011)

[15] Kästner, J.: Umbrella sampling. Wiley Interdisciplinary Reviews: Computational Molecular Science **1**(6), 932–942 (2011)

[16] Chodera, J.D., Noé, F.: Markov state models of biomolecular conformational dynamics. Current opinion in structural biology **25**, 135–144 (2014)

[17] Monticelli, L., Kandasamy, S.K., Periole, X., Larson, R.G., Tieleman, D.P., Marrink, S.-J.: The martini coarse-grained force field: extension to proteins. Journal of chemical theory and computation **4**(5), 819–834 (2008)

[18] Clementi, C.: Coarse-grained models of protein folding: toy models or predictive tools? Current opinion in structural biology **18**(1), 10–15

(2008)

[19] Wang, J., Olsson, S., Wehmeyer, C., Pérez, A., Charron, N.E., De Fabritiis, G., Noé, F., Clementi, C.: Machine learning of coarse-grained molecular dynamics force fields. ACS central science **5**(5), 755–767 (2019)

[20] Arts, M., Satorras, V.G., Huang, C.-W., Zuegner, D., Federici, M., Clementi, C., Noé, F., Pinsler, R., Berg, R.v.d.: Two for one: Diffusion models and force fields for coarse-grained molecular dynamics. arXiv preprint arXiv:2302.00600 (2023)

[21] Noé, F., Olsson, S., Köhler, J., Wu, H.: Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. Science **365**(6457), 1147 (2019)

[22] Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. Advances in neural information processing systems **31** (2018)

[23] Klein, L., Foong, A.Y., Fjelde, T.E., Mlodozeniec, B., Brockschmidt, M., Nowozin, S., Noé, F., Tomioka, R.: Timewarp: Transferable acceleration of molecular dynamics by learning time-coarsened dynamics. arXiv preprint arXiv:2302.01170 (2023)

[24] Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)

[25] Neal, R.M.: Annealed importance sampling. Statistics and computing **11**(2), 125–139 (2001)

[26] Del Moral, P., Doucet, A., Jasra, A.: Sequential Monte Carlo samplers. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **68**(3), 411–436 (2006)

[27] Doucet, A., Grathwohl, W.S., Matthews, A.G.d.G., Strathmann, H.: Annealed importance sampling meets score matching. In: ICLR Workshop on Deep Generative Models for Highly Structured Data (2022)

[28] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International Conference on Machine Learning, pp. 2256–2265 (2015). PMLR

[29] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems, vol. 33, pp. 6840–6851 (2020)

[30] Langevin, P.: Sur la théorie du mouvement brownien. Compt. Rendus **146**, 530–533 (1908)

[31] Uhlenbeck, G.E., Ornstein, L.S.: On the theory of the Brownian motion. Physical review **36**(5), 823 (1930)

[32] Roberts, G.O., Tweedie, R.L., *et al.*: Exponential convergence of Langevin distributions and their discrete approximations. Bernoulli **2**(4), 341–363 (1996)

[33] Wibisono, A., Wilson, A.C., Jordan, M.I.: A variational perspective on accelerated methods in optimization. proceedings of the National Academy of Sciences **113**(47), 7351–7358 (2016)

[34] Anderson, B.D.: Reverse-time diffusion equation models. Stochastic Processes and their Applications **12**(3), 313–326 (1982)

[35] Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: International Conference on Learning Representations (2021)

[36] Dhariwal, P., Nichol, A.: Diffusion models beat GANs on image synthesis. Advances in Neural Information Processing Systems **34**, 8780–8794 (2021)

[37] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 (2022)

[38] Risken, H.: Fokker-Planck equation. Springer (1996)

[39] Hutchinson, M.F.: A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. Communications in Statistics-Simulation and Computation **18**(3), 1059–1076 (1989)

[40] Grathwohl, W., Chen, R.T., Bettencourt, J., Sutskever, I., Duvenaud, D.: FFJORD: Free-form continuous dynamics for scalable reversible generative models. In: International Conference on Learning Representations (2019)

[41] Vincent, P.: A connection between score matching and denoising autoencoders. Neural computation **23**(7), 1661–1674 (2011)

[42] Alain, G., Bengio, Y.: What regularized auto-encoders learn from the data-generating distribution. The Journal of Machine Learning Research **15**(1), 3563–3593 (2014)

[43] Del Alamo, D., Sala, D., Mchaourab, H.S., Meiler, J.: Sampling alternative conformational states of transporters and receptors with alphafold2. Elife **11**, 75751 (2022)

[44] Zimmerman, M.I., Porter, J.R., Ward, M.D., Singh, S., Vithani, N., Meller, A., Mallimadugula, U.L., Kuhn, C.E., Borowsky, J.H., Wiewiora, R.P., *et al.*: Sars-cov-2 simulations go exascale to predict dramatic spike opening and cryptic pockets across the proteome. Nature chemistry **13**(7), 651–659 (2021)

[45] Zhang, L., Lin, D., Sun, X., Curth, U., Drosten, C., Sauerhering, L., Becker, S., Rox, K., Hilgenfeld, R.: Crystal structure of sars-cov-2 main protease provides a basis for design of improved α-ketoamide inhibitors. Science **368**(6489), 409–412 (2020)

[46] Tai, W., He, L., Zhang, X., Pu, J., Voronin, D., Jiang, S., Zhou, Y., Du, L.: Characterization of the receptor-binding domain (rbd) of 2019 novel coronavirus: implication for development of rbd protein as a viral attachment inhibitor and vaccine. Cellular & molecular immunology **17**(6), 613–620 (2020)

[47] Masureel, M., Martens, C., Stein, R.A., Mishra, S., Ruysschaert, J.-M., Mchaourab, H.S., Govaerts, C.: Protonation drives the conformational switch in the multidrug transporter lmrp. Nature chemical biology **10**(2), 149–155 (2014)

[48] Nussinov, R., Zhang, M., Liu, Y., Jang, H.: Alphafold, artificial intelligence (ai), and allostery. The Journal of Physical Chemistry B **126**(34), 6372–6383 (2022)

[49] Schindler, C.E., Baumann, H., Blum, A., Böse, D., Buchstaller, H.-P., Burgdorf, L., Cappel, D., Chekler, E., Czodrowski, P., Dorsch, D., *et al.*: Large-scale assessment of binding free energy calculations in active drug discovery projects. Journal of Chemical Information and Modeling **60**(11), 5457–5474 (2020)

[50] Wang, L., Wu, Y., Deng, Y., Kim, B., Pierce, L., Krilov, G., Lupyan, D., Robinson, S., Dahlgren, M.K., Greenwood, J., *et al.*: Accurate and reliable prediction of relative ligand binding potency in prospective drug discovery by way of a modern free-energy calculation protocol and force field. Journal of the American Chemical Society **137**(7), 2695–2703 (2015)

[51] Hafner, J.: Ab-initio simulations of materials using vasp: Density-functional theory and beyond. Journal of computational chemistry **29**(13), 2044–2078 (2008)

[52] Lu, Z.: Computational discovery of energy materials in the era of big data and machine learning: a critical review. Materials Reports: Energy **1**(3), 100047 (2021)

[53] Lu, Z.: Autonomous exploration and learning the off-equilibrium materials space for large-scale machine learning force fields. In preparation (2023)

[54] Ong, S.P., Richards, W.D., Jain, A., Hautier, G., Kocher, M., Cholia, S., Gunter, D., Chevrier, V.L., Persson, K.A., Ceder, G.: Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. Computational Materials Science **68**, 314–319 (2013)

[55] Durmus, A., Moulines, E.: High-dimensional Bayesian inference via the unadjusted Langevin algorithm. arXiv preprint arXiv:1605.01559 (2016)

[56] Cheng, X., Bartlett, P.: Convergence of Langevin MCMC in KL-divergence. arXiv preprint arXiv:1705.09048 (2017)

[57] Dalalyan, A.S.: Theoretical guarantees for approximate sampling from smooth and log-concave densities. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **79**(3), 651–676 (2017)

[58] Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics **378**, 686–707 (2019)

[59] Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. Machine learning **37**(2), 183–233 (1999)

[60] Wainwright, M.J., Jordan, M.I., *et al.*: Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning **1**(1–2), 1–305 (2008)

[61] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Proceedings of the International Conference on Learning Representations (ICLR 2014), Banff, Canada (2014). ICLR Committee

[62] Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: Proceedings of The 32nd International Conference on Machine Learning (ICML 2015), Lille, France, pp. 1530–1538 (2015). IMLS

[63] Kingma, D.P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., Welling, M.: Improved variational inference with inverse autoregressive flow. In: Advances in Neural Information Processing Systems, Barcelona,

Spain, pp. 4743–4751 (2016). NIPS Foundation

[64] Li, Y., Turner, R.E.: Renyi divergence variational inference. Advances in neural information processing systems **29** (2016)

[65] Hernandez-Lobato, J., Li, Y., Rowland, M., Bui, T., Hernandez-Lobato, D., Turner, R.: Black-box alpha divergence minimization. In: International Conference on Machine Learning, pp. 1511–1520 (2016). PMLR

[66] Midgley, L.I., Stimper, V., Simm, G.N., Schölkopf, B., Hernández-Lobato, J.M.: Flow annealed importance sampling bootstrap. arXiv preprint arXiv:2208.01893 (2022)

[67] Hyvärinen, A., Dayan, P.: Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research **6**(4) (2005)

[68] Cappé, O., Moulines, E., Rydén, T.: Inference in hidden Markov models (2005)

[69] Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. arXiv preprint arXiv:2206.00364 (2022)

[70] Song, Y., Durkan, C., Murray, I., Ermon, S.: Maximum likelihood training of score-based diffusion models. In: Advances in Neural Information Processing Systems, vol. 34, pp. 1415–1428 (2021)

[71] Lu, C., Zheng, K., Bao, F., Chen, J., Li, C., Zhu, J.: Maximum likelihood training for score-based diffusion ODEs by high order denoising score matching. In: International Conference on Machine Learning, pp. 14429–14460 (2022). PMLR

[72] Leach, A., Schmon, S.M., Degiacomi, M.T., Willcocks, C.G.: Denoising diffusion probabilistic models on SO(3) for rotational alignment. In: ICLR 2022 Workshop on Geometrical and Topological Representation Learning (2022)

[73] Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. In: Advances in Neural Information Processing Systems, vol. 32 (2019)

[74] Eastman, P., Swails, J., Chodera, J.D., McGibbon, R.T., Zhao, Y., Beauchamp, K.A., Wang, L.-P., Simmonett, A.C., Harrigan, M.P., Stern, C.D., *et al.*: Openmm 7: Rapid development of high performance algorithms for molecular dynamics. PLoS computational biology **13**(7), 1005659 (2017)

[75] Wang, R., Fang, X., Lu, Y., Yang, C.-Y., Wang, S.: The pdbbind database: methodologies and updates. Journal of medicinal chemistry **48**(12), 4111–4119 (2005)

[76] Rodríguez-Espigares, I., Torrens-Fontanals, M., Tiemann, J.K., Aranda-García, D., Ramírez-Anguita, J.M., Stepniewski, T.M., Worp, N., Varela-Rial, A., Morales-Pastor, A., Medel-Lacruz, B., *et al.*: Gpcrmd uncovers the dynamics of the 3d-gpcrome. Nature Methods **17**(8), 777–787 (2020)

[77] Min, Y., Wei, Y., Wang, P., Wu, N., Bauer, S., Zheng, S., Shi, Y., Wang, Y., Wang, X., Zhao, D., et al.: Predicting the protein-ligand affinity from molecular dynamics trajectories. arXiv preprint arXiv:2208.10230 (2022)

[78] He, J., Tian, K., Luo, S., Min, Y., Zheng, S., Shi, Y., He, D., Liu, H., Yu, N., Wang, L., et al.: Masked molecule modeling: A new paradigm of molecular representation learning for chemistry understanding (2022)

[79] Francoeur, P.G., Masuda, T., Sunseri, J., Jia, A., Iovanisci, R.B., Snyder, I., 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)

[80] Shi, Y., Zheng, S., Ke, G., Shen, Y., You, J., He, J., Luo, S., Liu, C., He, D., Liu, T.-Y.: Benchmarking graphormer on large-scale molecular modeling datasets. arXiv preprint arXiv:2203.04810 (2022)

[81] Ho, J., Salimans, T.: Classifier-free diffusion guidance. arXiv preprint arXiv:2207.12598 (2022)

[82] Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2021)

[83] Bao, F., Li, C., Zhu, J., Zhang, B.: Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In: International Conference on Learning Representations (2022)

[84] Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In: Advances in Neural Information Processing Systems

[85] Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. arXiv preprint arXiv:2211.01095 (2022)

[86] Karras, T., Aittala, M., Aila, T., Laine, S.: Elucidating the design space of diffusion-based generative models. In: Advances in Neural Information

Processing Systems (2022)

[87] Perez-Hernandez, G., Paul, F., Giorgino, T., De Fabritiis, G., Noé, F.: Identification of slow molecular order parameters for markov model construction. The Journal of chemical physics **139**(1) (2013)

[88] Schwantes, C.R., Pande, V.S.: Improvements in markov state model construction reveal many non-native interactions in the folding of ntl9. Journal of chemical theory and computation **9**(4), 2000–2009 (2013)

[89] Scherer, M.K., Trendelkamp-Schroer, B., Paul, F., Pérez-Hernández, G., Hoffmann, M., Plattner, N., Wehmeyer, C., Prinz, J.-H., Noé, F.: PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. Journal of Chemical Theory and Computation **11**, 5525–5542 (2015). https://doi.org/10.1021/acs.jctc.5b00743. Accessed 2015-10-19

[90] McGibbon, R.T., Beauchamp, K.A., Harrigan, M.P., Klein, C., Swails, J.M., Hernández, C.X., Schwantes, C.R., Wang, L.-P., Lane, T.J., Pande, V.S.: Mdtraj: A modern open library for the analysis of molecular dynamics trajectories. Biophysical Journal **109**(8), 1528–1532 (2015). https://doi.org/10.1016/j.bpj.2015.08.015

[91] Zhang, Y., Skolnick, J.: Scoring function for automated assessment of protein structure template quality. Proteins: Structure, Function, and Bioinformatics **57**(4), 702–710 (2004)

[92] Xu, J., Zhang, Y.: How significant is a protein structure similarity with tm-score= 0.5? Bioinformatics **26**(7), 889–895 (2010)

[93] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[94] Jing, B., Eismann, S., Suriana, P., Townshend, R.J., Dror, R.: Learning from protein structure with geometric vector perceptrons. arXiv preprint arXiv:2009.01411 (2020)

[95] Schütt, K., Unke, O., Gastegger, M.: Equivariant message passing for the prediction of tensorial properties and molecular spectra. In: International Conference on Machine Learning, pp. 9377–9388 (2021). PMLR

[96] wwPDB Consortium: Protein Data Bank: the single global archive for 3d macromolecular structure data. Nucleic acids research **47**(D1), 520–528 (2019)

[97] Steinegger, M., Söding, J.: Clustering huge protein sequence sets in linear

time. Nature communications **9**(1), 2542 (2018)

[98] Zhang, S., Krieger, J.M., Zhang, Y., Kaya, C., Kaynak, B., Mikulska-Ruminska, K., Doruker, P., Li, H., Bahar, I.: Prody 2.0: increased scale and scope after 10 years of protein dynamics modelling with python. Bioinformatics **37**(20), 3657–3659 (2021)

[99] Eastman, P., Friedrichs, M.S., Chodera, J.D., Radmer, R.J., Bruns, C.M., Ku, J.P., Beauchamp, K.A., Lane, T.J., Wang, L.-P., Shukla, D., *et al.*: Openmm 4: a reusable, extensible, hardware independent library for high performance molecular simulation. Journal of chemical theory and computation **9**(1), 461–469 (2013)

[100] Su, M., Yang, Q., Du, Y., Feng, G., Liu, Z., Li, Y., Wang, R.: Comparative assessment of scoring functions: the casf-2016 update. Journal of chemical information and modeling **59**(2), 895–913 (2018)

[101] Van Der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A.E., Berendsen, H.J.: Gromacs: fast, flexible, and free. Journal of computational chemistry **26**(16), 1701–1718 (2005)

[102] Lindorff-Larsen, K., Piana, S., Palmo, K., Maragakis, P., Klepeis, J.L., Dror, R.O., Shaw, D.E.: Improved side-chain torsion potentials for the amber ff99sb protein force field. Proteins: Structure, Function, and Bioinformatics **78**(8), 1950–1958 (2010)

[103] Sousa da Silva, A.W., Vranken, W.F.: Acpype-antechamber python parser interface. BMC research notes **5**(1), 1–8 (2012)

[104] Van Gunsteren, W.F., Berendsen, H.J.: A leap-frog algorithm for stochastic dynamics. Molecular Simulation **1**(3), 173–185 (1988)

[105] Essmann, U., Perera, L., Berkowitz, M.L., Darden, T., Lee, H., Pedersen, L.G.: A smooth particle mesh ewald method. The Journal of chemical physics **103**(19), 8577–8593 (1995)

[106] Hess, B., Bekker, H., Berendsen, H.J., Fraaije, J.G.: Lincs: a linear constraint solver for molecular simulations. Journal of computational chemistry **18**(12), 1463–1472 (1997)

[107] Mongan, J., Case, D.A., McCammon, J.A.: Constant ph molecular dynamics in generalized born implicit solvent. Journal of computational chemistry **25**(16), 2038–2048 (2004)

[108] Hammer, B., Hansen, L.B., Nørskov, J.K.: Improved adsorption energetics within density-functional theory using revised perdew-burke-ernzerhof functionals. Physical review B **59**(11), 7413 (1999)

[109] Kresse, G., Furthmüller, J.: Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. Physical review B **54**(16), 11169 (1996)

[110] Ramachandran, G.N., Ramakrishnan, C., Sasisekharan, V.: Stereochemistry of polypeptide chain configurations. Journal of Molecular Biology **7**(1), 95–99 (1963). https://doi.org/10.1016/S0022-2836(63)80023-6

[111] Köhler, J., Chen, Y., Kramer, A., Clementi, C., Noé, F.: Flow-Matching: Efficient coarse-graining of molecular dynamics without forces. Journal of Chemical Theory and Computation **19**(3), 942–952 (2023)

5 Acknowledgements

We thank Nathan A. Baker, Lixin Sun, Bas Veeling, Victor García Satorras, Andrew Foong and Cheng Lu for insightful discussions; Shengjie Luo for helping with dataset preparations; Jingjie Su for managing the project; Jingyun Bai for helping with figure design; colleagues at Microsoft for their encouragement and support.

6 Author information

Contributions

S.Zheng and TY.Liu led the research. S.Zheng, J.He, C.Liu, Z.Lu and H.Liu conceived the project. J.He, C.Liu, Y.Shi, W.Feng and F.Ju and J.Wang developed the diffusion model and training pipeline. J.He, Y.Shi, Z.Lu, J.Zhu, F.Ju, H.Zhang and H.Liu developed data and analytics systems. H.Liu, Y.Shi, Z.Lu, Y.Min and S.Tang conducted simulations. H.Hao, P.Jin, C.Chen, and F.Noé contributed technical advice and ideas. S.Zheng, J.He, C.Liu, Y.Shi, Z.Lu, F.Noé, H.Zhang and H.Liu wrote the paper with the inputs from all authors.

Corresponding authors

Correspondence to Shuxin Zheng, Chang Liu, Haiguang Liu and Tie-Yan Liu.

Appendix A Technical Details

A.1 Formulation of DiG

The forward process Eq. (1) is constructed using the Langevin dynamics that takes the simple distribution $p_{\text{simple}} := \mathcal{N}(\mathbf{0}, \mathbf{I})$ as its stationary distribution: $\mathrm{d}\mathbf{R}_t = \frac{1}{2}\nabla \log p_{\text{simple}}(\mathbf{R}_t)\,\mathrm{d}t + \mathrm{d}\mathbf{B}_t = -\frac{1}{2}\mathbf{R}_t\,\mathrm{d}t + \mathrm{d}\mathbf{B}_t$. From any conformational distribution of any system as the initial distribution $q_{\mathcal{D},0}(\mathbf{R}_0)$, the distribution evolves under this process and converges to p_{simple} exponentially [32, 55–57]. For a faster simulation convergence, it is preferred to introduce a time dilation scheme β_t that increases in t [33]. This gives Eq. (1).

To draw structure samples using DiG, we simulate the reverse process Eq. (2) from samples from p_{simple} the standard Gaussian distribution, which can be easily drawn independently. Note that this "reverse" is not a sample-level point-to-point inverse, but a distribution-level inverse: denoting its induced distribution as $p_{\bar{t}}$, if $p_{\bar{t}=0} = q_{\mathcal{D},\tau}$, then $p_{\bar{t}=\tau} = q_{\mathcal{D},0}$. When employing a trained score model $\mathbf{s}^{\theta}_{\mathcal{D},t}$ to approximate the score function $\nabla \log q_{\mathcal{D},t}$, the reverse process can be simulated by the Euler-Maruyama discretization using the step size h up to $o(h)$ local error: $\mathbf{R}_{\bar{i}+1} = \left(1 + \frac{1}{2}\beta_{\bar{i}}\right)\mathbf{R}_{\bar{i}} + \beta_{\bar{i}}\mathbf{s}^{\theta}_{\mathcal{D},\bar{i}}(\mathbf{R}_{\bar{i}}) + \mathcal{N}(\mathbf{0}, \beta_{\bar{i}}\mathbf{I})$, where the indexed quantities are evaluated at $\bar{t} := \bar{i}h$, and $\beta_{\bar{i}} := h\beta_{\bar{t}=\bar{i}h}$, and "$+\mathcal{N}(\cdots)$" denotes adding a randomly drawn sample from the denoted Gaussian distribution. In the original step index i, this becomes: $\mathbf{R}_{i-1} = \left(1 + \frac{1}{2}\beta_i\right)\mathbf{R}_i + \beta_i \mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i) + \mathcal{N}(\mathbf{0}, \beta_i \mathbf{I})$. By design h is chosen small for accurate simulation, so is each β_i. Hence we can leverage the approximation $\frac{1}{\sqrt{1-\beta_i}} = 1 + \frac{1}{2}\beta_i + o(h)$ which does not increase the dicretization local error. This gives Eq. (3) for generating samples from the equilibrium distribution.

A.1.1 Physics-Informed Diffusion Pre-training

The goal of the score model $\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)$ is to match the corresponding true score function $\nabla \log q_{\mathcal{D},t}(\mathbf{R}_t)$ from the forward process Eq. (1) for each $t \in [0, \tau]$. To better leverage the diffusion-process construction of DiG, we use a partial differential equation governing the true score function and construct a loss function that enforces the equation to hold for training the score model.

Under a general diffusion process $\mathrm{d}\mathbf{R}_t = \mathbf{f}_t(\mathbf{R}_t)\,\mathrm{d}t + g_t\,\mathrm{d}\mathbf{B}_t$, the instantaneous distribution transformation is given by the Fokker-Planck equation (FPE; in logarithm form):

$$\begin{aligned}\frac{\partial}{\partial t}\log q_t(\mathbf{R}_t) = &-\nabla\cdot\mathbf{f}_t(\mathbf{R}_t) - \nabla \log q_t(\mathbf{R}_t)\cdot\mathbf{f}_t(\mathbf{R}_t)\\ &+\frac{g_t^2}{2}\left(\nabla^2 \log q_t(\mathbf{R}_t) + \|\nabla \log q_t(\mathbf{R}_t)\|^2\right),\end{aligned} \tag{A1}$$

Table A1: Notations.

	General formulation
$\mathcal{D}$	System descriptor
$\mathbf{R}$	Molecular structure
$\{\mathbf{R}_{\mathcal{D},0}^{(m)}\}_{m=1}^{M}$	Molecular structures of system $\mathcal{D}$ in the dataset for PIDP training
$\{\mathbf{R}_{\mathcal{D},0}^{(n)}\}_{n=1}^{N_{\text{data}}}$	Molecular structures of system $\mathcal{D}$ in the dataset for data-based (denoising score matching) training
D	Dimension of $\mathbf{R}$
$E_{\mathcal{D}}(\mathbf{R})$	(Potential) Energy function of system $\mathcal{D}$
k_{B}	Boltzmann constant
T	Temperature
c	A property of molecular structure
I	Length of descriptor / number of individual elements in a system
$\imath, \jmath \in \{1, 2, \cdots, I\}$	Index for individual elements in a system
	Diffusion process
τ	Total time length/duration of the forward diffusion process
$t \in [0, \tau]$	Time variable (continuous)
N	Number of time discretization steps for the diffusion process
$i \in \{1, 2, \cdots, N\}$	Time step (discrete)
$h\ (= \tau/N)$	Time discretization step size
$\bar{t}$ or $\bar{i}$	Reverse time or step
$\mathbf{B}_t$ and $\bar{\mathbf{B}}_t$	Standard Brownian motion in dimension D and its reverse process
$\mathbf{f}_t(\mathbf{R}_t)$	Drift function in a general diffusion process
g_t	Diffusion rate scheme in a general diffusion process
$q_{\mathcal{D},0}$	Equilibrium distribution of system $\mathcal{D}$ (under a certain temperature)
$\mathbf{R}_0$	Molecular structure variable following equilibrium distribution $q_{\mathcal{D},0}$
$q_{\mathcal{D},t}$ or $q_{\mathcal{D},i}$	Distribution of molecular structure in intermediate time or step in the forward diffusion process
$\mathbf{R}_t$ or $\mathbf{R}_i$	Molecular structure variable in intermediate time or step, following $q_{\mathcal{D},t}$ or $q_{\mathcal{D},i}$
$q(\mathbf{R}_t \mid \mathbf{R}_0)$ or $q(\mathbf{R}_i \mid \mathbf{R}_0)$	Marginal transition kernel of the forward diffusion process
β_t or β_i	Time dilation scheme. Note $\beta_i := h\beta_{t_i}$ which is different from others
σ_t or σ_i	Noise variance scheme (standard deviation of $q(\mathbf{R}_t \mid \mathbf{R}_0)$ or $q(\mathbf{R}_i \mid \mathbf{R}_0)$)
ϵ_t or ϵ_i	Standard Gaussian noise variable
$\mathbf{s}_{\mathcal{D},t}^{\theta}$ or $\mathbf{s}_{\mathcal{D},i}^{\theta}$	Score model for $q_{\mathcal{D},t}$ or $q_{\mathcal{D},i}$
$\epsilon_{\mathcal{D},t}^{\theta}$ or $\epsilon_{\mathcal{D},i}^{\theta}$	Noise-predicting model for $q_{\mathcal{D},t}$ or $q_{\mathcal{D},i}$
p_{simple}	The simple distribution to which the forward diffusion process converges
$p_{\mathcal{D},t}^{\theta}$ or $p_{\mathcal{D},i}^{\theta}$	Distribution of molecular structure in intermediate time or step in the reverse diffusion process simulated by $\mathbf{s}_{\mathcal{D},t}^{\theta}$ or $\mathbf{s}_{\mathcal{D},i}^{\theta}$ or $\epsilon_{\mathcal{D},t}^{\theta}$ or $\epsilon_{\mathcal{D},i}^{\theta}$

For the specific diffusion process Eq. (1), the evolving distribution q_t from the forward process satisfies: $\frac{\partial}{\partial t}\log q_{\mathcal{D},t}(\mathbf{R}_t) = \frac{\beta_t}{2}\Big(D + \mathbf{R}_t \cdot \nabla \log q_{\mathcal{D},t}(\mathbf{R}_t) + \nabla^2 \log q_{\mathcal{D},t}(\mathbf{R}_t) + \|\nabla \log q_{\mathcal{D},t}(\mathbf{R}_t)\|^2\Big)$, where D is the dimension of $\mathbf{R}$. Taking the gradient of the above equation gives: $\frac{\partial}{\partial t}\nabla \log q_{\mathcal{D},t}(\mathbf{R}_t) = \frac{\beta_t}{2}\Big(\nabla \cdot \big(\mathbf{R}_t \cdot \nabla \log q_{\mathcal{D},t}(\mathbf{R}_t)\big) + \nabla\big(\nabla \cdot \nabla \log q_{\mathcal{D},t}(\mathbf{R}_t)\big) + \nabla\|\nabla \log q_{\mathcal{D},t}(\mathbf{R}_t)\|^2\Big)$, which becomes an equation of the score function $\nabla \log q_{\mathcal{D},t}(\mathbf{R}_t)$. To well approximate $\nabla \log q_{\mathcal{D},t}(\mathbf{R}_t)$, the score model $\mathbf{s}^\theta_{\mathcal{D},t}(\mathbf{R}_t)$ also needs to satisfy this equation. To enforce it, we follow the idea of physics-informed neural networks [58] that converts a differential equation into a loss function of the to-be-solved function. The loss function is typically taken as the squared norm of the equality residual, which in our case is:

$$\left\|\frac{\beta_t}{2}\Big(\nabla \cdot \big(\mathbf{R}_t \cdot \mathbf{s}^\theta_{\mathcal{D},t}(\mathbf{R}_t)\big) + \nabla\big(\nabla \cdot \mathbf{s}^\theta_{\mathcal{D},t}(\mathbf{R}_t)\big) + \nabla\big\|\mathbf{s}^\theta_{\mathcal{D},t}(\mathbf{R}_t)\big\|^2\Big) - \frac{\partial}{\partial t}\mathbf{s}^\theta_{\mathcal{D},t}(\mathbf{R}_t)\right\|^2,$$

for each t. By time discretization and evaluating the loss on a set of samples $\{\mathbf{R}^{(m)}_{\mathcal{D},i}\}_{m=1}^M$, this gives the first term in Eq. (4).

The FPE does not have (nor need) a boundary condition as long as each $q_{\mathcal{D},t}$ is normalized. For the initial condition, we know that the score of the target equilibrium distribution $\nabla \log q_{\mathcal{D},0}(\mathbf{R}_0) = -\nabla E_{\mathcal{D}}(\mathbf{R}_0)/(k_{\mathrm{B}}T)$ is exactly given by the gradient of the energy function of the system. This is where the energy function comes to supervise the model, and this supervision propagates to other time steps via the first term of the loss. To implement this initial condition, we minimize $\big\|\mathbf{s}^\theta_0(\mathbf{R}_0) - \nabla \log q_{\mathcal{D},0}(\mathbf{R}_0)\big\|^2 = \big\|\mathbf{s}^\theta_0(\mathbf{R}_0) + \nabla E_{\mathcal{D}}(\mathbf{R}_0)/(k_{\mathrm{B}}T)\big\|^2$, which leads to the second term in Eq. (4). Note that in Eq. (4) the loss term is not imposed on $t_0 = 0$ (i.e., $i = 0$). This is because in the actual implementation, the score model is expressed using a noise-predicting model $\epsilon^\theta_{\mathcal{D},t}$ as $\mathbf{s}^\theta_{\mathcal{D},t} = -\sigma_t \epsilon^\theta_{\mathcal{D},t}$ (explained in Supplementary Sec. A.1.2), which, at $t = 0$, the vanishing $\sigma_0 = 0$ causes an ill-defined score model. This is commonly solved by starting the diffusion simulation from an infinitesimal initial time step [35], which corresponds to $t = h$ or $i = 1$ here. On the other hand, from the data-generation process Eq. (3), the last required time step for the model is $i = 1$, where the sample needs to be updated to follow the equilibrium distribution. So it is reasonable to supervise $\mathbf{s}^\theta_{\mathcal{D},i=1}$ or $\epsilon^\theta_{\mathcal{D},i=1}$ with the energy function.

In comparison, we note that there are other common approaches to train a generative model using a given energy function, but they cannot leverage the advantage of the diffusion-process construction of DiG and thus do not enjoy the step-by-step supervision pattern and are not as effective to train large models. The most popular way is to minimize the reverse Kullback-Leibler (KL) divergence $\mathrm{KL}(p^\theta_{\mathcal{D},0}\|q_{\mathcal{D},0})$ between the model-defined equilibrium distribution and the true equilibrium distribution, which is equivalent to minimizing the

(Helmholtz) free energy:

$$\text{FreeEng}_{\mathcal{D}} = \mathbb{E}_{p^{\theta}_{\mathcal{D},0}(\mathbf{R}_0)}[E_{\mathcal{D}}(\mathbf{R}_0) + k_{\text{B}}T \log p^{\theta}_{\mathcal{D},0}(\mathbf{R}_0)].$$

In the expression, no sample from $q_{\mathcal{D},0}$ is required, so the access to the energy function $E_{\mathcal{D}}$ suffices for training. This approach is known as variational inference [59–63] in machine learning, and the negative (Helmholtz) free energy is also called evidence lower bound (ELBO). This method is recently used to train a generative model for the equilibrium distribution of molecular systems [21]. A more modern approach minimizes the alpha divergence between the model and the equilibrium distributions [64, 65], which generalizes the reverse KL divergence and ameliorates the mode-collapse tendency to some extent. It is also applied to molecular systems recently [66]. These methods can be directly applied to DiG given the density evaluation method Eq. (6), but it loses step-by-step supervision as it only supervises the end distribution $p^{\theta}_{\mathcal{D},0}$, which makes training large models hard. Moreover, evaluating the density function requires an ODE solver, so the optimization requires backpropagation through the ODE solver, which is very costly.

A.1.2 Training DiG with Data

To develop a method to train the model step-by-step using data from $q_{\mathcal{D},0}$, we start by score matching for each step i, that is to minimize

$$\mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \left\| \mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i) - \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2.$$

Although this loss can be made tractable (i.e., to get rid of the unknown true score function $\nabla \log q_{\mathcal{D},i}$) using the standard score matching technique [67], the resulting loss function involves the divergence of the score model $\nabla \cdot \mathbf{s}^{\theta}_{\mathcal{D},i}$ which is expensive to evaluate and optimize. Another way to make it tractable is via the denoising score matching technique [41, 42]. The method first reforms the intermediate marginal distribution in terms of the marginal transition kernel $q(\mathbf{R}_i \mid \mathbf{R}_0)$ from the forward process (which does not depend on a specific system hence no $\mathcal{D}$ subscript), $q_{\mathcal{D},i}(\mathbf{R}_i) = \int q_{\mathcal{D},0}(\mathbf{R}_0) q(\mathbf{R}_i \mid \mathbf{R}_0) \, \mathrm{d}\mathbf{R}_0$, and then decompose the score function as:

$$\begin{aligned} \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) &= \frac{1}{q_{\mathcal{D},i}(\mathbf{R}_i)} \int q_{\mathcal{D},0}(\mathbf{R}_0) \nabla_{\mathbf{R}_i} q(\mathbf{R}_i \mid \mathbf{R}_0) \, \mathrm{d}\mathbf{R}_0 \\ &= \int q_{\mathcal{D},0}(\mathbf{R}_0) \frac{q(\mathbf{R}_i \mid \mathbf{R}_0)}{q_{\mathcal{D},i}(\mathbf{R}_i)} \nabla_{\mathbf{R}_i} \log q(\mathbf{R}_i \mid \mathbf{R}_0) \, \mathrm{d}\mathbf{R}_0 \\ &= \mathbb{E}_{q_{\mathcal{D}}(\mathbf{R}_0 \mid \mathbf{R}_i)} [\nabla_{\mathbf{R}_i} \log q(\mathbf{R}_i \mid \mathbf{R}_0)], \end{aligned} \tag{A2}$$

a.k.a Fisher's identity [68]. The score-matching loss then becomes:

$$\mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \left\| \mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i) - \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2$$

$$
\begin{aligned}
&= \mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \left\| \mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2 - 2\mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \left[\mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) \cdot \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) \right] \\
&\quad + \mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \left\| \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2 \\
&\overset{\text{Eq. (A2)}}{=} \mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \mathbb{E}_{q(\mathbf{R}_0|\mathbf{R}_i)} \left\| \mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2 \\
&\quad - 2\mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \mathbb{E}_{q(\mathbf{R}_0|\mathbf{R}_i)} \left[\mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) \cdot \nabla_{\mathbf{R}_i} \log q(\mathbf{R}_i \mid \mathbf{R}_0) \right] \\
&\quad + \mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \left\| \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2 \\
&= \mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \mathbb{E}_{q(\mathbf{R}_0|\mathbf{R}_i)} \left\| \mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) - \nabla_{\mathbf{R}_i} \log q(\mathbf{R}_i \mid \mathbf{R}_0) \right\|^2 \\
&\quad - \mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \mathbb{E}_{q(\mathbf{R}_0|\mathbf{R}_i)} \left\| \nabla_{\mathbf{R}_i} \log q(\mathbf{R}_i \mid \mathbf{R}_0) \right\|^2 \\
&\quad + \mathbb{E}_{q_{\mathcal{D},i}(\mathbf{R}_i)} \left\| \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2 \\
&= \mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)} \mathbb{E}_{q(\mathbf{R}_i|\mathbf{R}_0)} \left\| \mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) - \nabla_{\mathbf{R}_i} \log q(\mathbf{R}_i \mid \mathbf{R}_0) \right\|^2 \\
&\quad + \mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)} \mathbb{E}_{q(\mathbf{R}_i|\mathbf{R}_0)} \left[\left\| \nabla \log q_{\mathcal{D},i}(\mathbf{R}_i) \right\|^2 - \left\| \nabla_{\mathbf{R}_i} \log q(\mathbf{R}_i \mid \mathbf{R}_0) \right\|^2 \right].
\end{aligned}
$$

Noting that the second term in the last expression is a constant of θ, optimizing the score-matching loss for step i is equivalent to minimizing the first term:

$$
\mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)} \mathbb{E}_{q(\mathbf{R}_i|\mathbf{R}_0)} \left\| \mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) - \nabla_{\mathbf{R}_i} \log q(\mathbf{R}_i \mid \mathbf{R}_0) \right\|^2. \tag{A3}
$$

This is the denoising score matching loss. To explain the name, in the original context, $q(\mathbf{R}_i \mid \mathbf{R}_0) = \mathcal{N}(\mathbf{R}_i \mid \mathbf{R}_0, \sigma_i^2 \mathbf{I})$ which adds noise to the data sample $\mathbf{R}_0$ to get a noisy version $\mathbf{R}_i$, and the resulting loss

$$
\begin{aligned}
&\mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)} \mathbb{E}_{q(\mathbf{R}_i|\mathbf{R}_0)} \left\| \mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) + \frac{\mathbf{R}_i - \mathbf{R}_0}{\sigma_i^2} \right\|^2 \\
&= \frac{1}{\sigma_i^2} \mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)} \mathbb{E}_{q(\mathbf{R}_i|\mathbf{R}_0)} \left\| \mathbf{R}_i + \sigma_i^2 \mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i) - \mathbf{R}_0 \right\|^2
\end{aligned}
$$

drives the "decoder" $\mathbf{R}_i + \sigma_i^2 \mathbf{s}^\theta_{\mathcal{D},i}(\mathbf{R}_i)$ to recover the original clean data point $\mathbf{R}_0$ by "denoising" $\mathbf{R}_i$.

Optimizing the denoising score matching loss Eq. (A3) is tractable once we know the conditional distribution $q(\mathbf{R}_i \mid \mathbf{R}_0)$, which is fortunately available in closed form for the forward process Eq. (1). Under continuous-time, the result is $q(\mathbf{R}_t \mid \mathbf{R}_0) = \mathcal{N}(\mathbf{R}_t \mid \alpha_t \mathbf{R}_0, \sigma_t^2 \mathbf{I})$ [35, 69], where $\alpha_t := \exp(-\frac{1}{2}\int_0^t \beta_{t'} \, \mathrm{d}t')$ and $\sigma_t := \sqrt{1 - \alpha_t^2}$. For a discretized expression, recall that the time interval $[0, \tau]$ is uniformly divided into $N + 1$ points with step size $h = \tau / N$, step i corresponds to time $t = ih$, and $\beta_i := h\beta_{t=ih}$. This leads to

$$
\alpha_{t=ih} = \sqrt{\exp(-\sum_{j=1}^{i} \beta_j + o(h))} = \sqrt{\exp(o(h)) \prod_{j=1}^{i} \exp(-\beta_j)}
$$

$$= \sqrt{(1+o(h))\prod_{j=1}^{i}(1-\beta_j+o(h))} = \sqrt{\prod_{j=1}^{i}(1-\beta_j)+o(h)}$$
$$= \prod_{j=1}^{i}\sqrt{1-\beta_j}+o(h),$$

so we can take $\alpha_i := \prod_{j=1}^{i}\sqrt{1-\beta_j}$. Correspondingly, $\sigma_i = \sqrt{1-\alpha_i^2}$. The required conditional distribution is then:

$$q(\mathbf{R}_i \mid \mathbf{R}_0) = \mathcal{N}(\mathbf{R}_i \mid \alpha_i \mathbf{R}_0, \sigma_i^2 \mathbf{I}). \tag{A4}$$

The loss Eq. (A3) for time step i then becomes $\mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)}\mathbb{E}_{q(\mathbf{R}_i|\mathbf{R}_0)}\left\|\mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i) + \frac{1}{\sigma_i^2}(\mathbf{R}_i - \alpha_i\mathbf{R}_0)\right\|^2$. Using the reparameterization of the Gaussian distribution $q(\mathbf{R}_i \mid \mathbf{R}_0)$ as $\mathbf{R}_i = \alpha_i\mathbf{R}_0 + \sigma_i\epsilon_i$ where $\epsilon_i \sim p(\epsilon_i) := \mathcal{N}(\mathbf{0}, \mathbf{I})$, the loss is further reformed as: $\mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)}\mathbb{E}_{p(\epsilon_i)}\left\|\mathbf{s}^{\theta}_{\mathcal{D},i}(\alpha_i\mathbf{R}_0 + \sigma_i\epsilon_i) + \frac{1}{\sigma_i}\epsilon_i\right\|^2 = \frac{1}{\sigma_i^2}\mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)}\mathbb{E}_{p(\epsilon_i)}\left\|\sigma_i\mathbf{s}^{\theta}_{\mathcal{D},i}(\alpha_i\mathbf{R}_0 + \sigma_i\epsilon_i) + \epsilon_i\right\|^2$. To balance the scale of the loss for different $i \in \{1, \cdots, N\}$, the loss Eq. (A3) for step i is normalized by the scale of $\mathbb{E}_{q(\mathbf{R}_i|\mathbf{R}_0)}\|\nabla_{\mathbf{R}_i}\log q(\mathbf{R}_i \mid \mathbf{R}_0)\|^2 = \mathbb{E}_{p(\epsilon_i)}\left\|\frac{\epsilon_i}{\sigma_i}\right\|^2 = \frac{1}{\sigma_i^2}$ [35], which finally leads to Eq. (5).

From the expression of this loss Eq. (5), we find that the "model" $-\sigma_i\mathbf{s}^{\theta}_{\mathcal{D},i}(\alpha_i\mathbf{R}_0 + \sigma_i\epsilon_i)$ can be seen as to "predict the noise label" ϵ_i, whose distribution is well centered and scaled. This is the range that a deep learning model works the best. So to make a comfortable and friendly learning task, we implement the model to directly output the vector value for $-\sigma_i\mathbf{s}^{\theta}_{\mathcal{D},i}$, which we denote as $\epsilon^{\theta}_{\mathcal{D},i}$ and call it the noise-predicting model. The score model can still be recovered by:

$$\mathbf{s}^{\theta}_{\mathcal{D},i}(\mathbf{R}_i) = -\epsilon^{\theta}_{\mathcal{D},i}(\mathbf{R}_i)/\sigma_i, \tag{A5}$$

as an approximation to the true score function $\nabla \log q_{\mathcal{D},i}(\mathbf{R}_i)$. The training loss Eq. (5) then becomes:

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{q_{\mathcal{D},0}(\mathbf{R}_0)}\mathbb{E}_{p(\epsilon_i)}\left\|\epsilon^{\theta}_{\mathcal{D},i}(\alpha_i\mathbf{R}_0 + \sigma_i\epsilon_i) - \epsilon_i\right\|^2. \tag{A6}$$

This recovers the formulation in [29, 35]. To understand the loss, note the marginal transition kernel Eq. (A4) of the forward process means $\mathbf{R}_i = \alpha_i\mathbf{R}_0 + \sigma_i\epsilon_i$ where $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. So the $\epsilon^{\theta}_{\mathcal{D},i}$ model tries to recover the noise variable ϵ_i from $\mathbf{R}_i$ that were used to generate $\mathbf{R}_i$.

A.1.3 Density Evaluation using DiG

Viewed in the continuous-time limit, DiG defines a distribution via transforming p_{simple} through the reverse process Eq. (2), where the score function is approximated by the model. Written in forward time t, this process follows the following SDE:

$$\mathrm{d}\mathbf{R}_t = -\frac{\beta_t}{2}\mathbf{R}_t\,\mathrm{d}t - \beta_t \mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\,\mathrm{d}t + \sqrt{\beta_t}\,\mathrm{d}\bar{\mathbf{B}}_t, \tag{A7}$$

where $\bar{\mathbf{B}}_t$ is the reverse of the Brownian motion. The distribution transformation under this process is given by its FPE in Eq. (A1):

$$\begin{aligned}
&\frac{\partial}{\partial t}\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t) \\
=\ &-\nabla\cdot\Big(-\frac{\beta_t}{2}\mathbf{R}_t - \beta_t\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\Big) - \nabla\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\cdot\Big(-\frac{\beta_t}{2}\mathbf{R}_t - \beta_t\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\Big) \\
&-\frac{\beta_t}{2}\Big(\nabla^2\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t) + \big\|\nabla\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\big\|^2\Big),
\end{aligned} \tag{A8}$$

where the last term has a negative sign in correspondence to the reverse Brownian motion. When the model is well-learned, $\mathbf{s}^{\theta}_{\mathcal{D},t}$ well approximates $\nabla\log q_{\mathcal{D},t}$ and $p^{\theta}_{\mathcal{D},t}$ well approximates $q_{\mathcal{D},t}$, hence we can approximate $\nabla\log p^{\theta}_{\mathcal{D},t}$ also using $\mathbf{s}^{\theta}_{\mathcal{D},t}$. This turns Eq. (A8) into: [1]

$$\begin{aligned}
\frac{\partial}{\partial t}\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t) =\ &-\nabla\cdot\Big(-\frac{\beta_t}{2}\mathbf{R}_t - \frac{\beta_t}{2}\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\Big) \\
&-\nabla\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\cdot\Big(-\frac{\beta_t}{2}\mathbf{R}_t - \frac{\beta_t}{2}\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\Big).
\end{aligned} \tag{A9}$$

Comparing this equation with the general-form FPE in Eq. (A1), we can find that this equation is exactly the FPE of the "deterministic diffusion process" defined by the ODE in Eq. (7). In other words, the ODE in Eq. (7), and the SDE in Eq. (A7), render the same $\frac{\partial}{\partial t}\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)$ hence the same marginal distribution $p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)$ in each time step t (since they have the same terminal distribution $p_\tau = p_{\text{simple}}$; note the mentioned requirement $\mathbf{s}^{\theta}_{\mathcal{D},t} = \nabla\log p^{\theta}_{\mathcal{D},t}$ for this claim to hold). Since the SDE in Eq. (A7) is the same as Eq. (2) and in turn leads to the sampling/generation process in Eq. (3), this finding indicates that we can also generate equilibrium-distribution samples by simulating the ODE in Eq. (7). This kind of deterministic process or ODE sampling process is used in protein conformation sampling (see the end of Supplementary Sec. A.2.1) and property-guided structure generation (Supplementary Sec. A.5).

Back to density evaluation using DiG, we can estimate the density function of the model-defined equilibrium distribution $p^{\theta}_{\mathcal{D},0}$ by integrating w.r.t

[1] When $\mathbf{s}^{\theta}_t \neq \nabla\log q_{\mathcal{D},t}$ or $q_{\mathcal{D},\tau} \neq p_\tau := p_{\text{simple}}$, Eq. (A8) and Eq. (A9) (or Eq. (A7) and Eq. (7)) give different evolving densities. See [70, 71] for more discussions.

the diffusion time step t following the above ODE in Eq. (A9), which does not contain any unknown objects (recall that we made the assumption that $\mathbf{s}^{\theta}_{\mathcal{D},t} = \nabla \log p^{\theta}_{\mathcal{D},t}$ to use Eq. (A9)). Let $\mathbf{R}_t$ be a solution to Eq. (7), which is a deterministic curve in the state space. Then we find the total derivative w.r.t time t (a.k.a material/particle derivative) is:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t) &= \frac{\partial}{\partial t}\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t) + \nabla \log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\cdot\frac{\mathrm{d}\mathbf{R}_t}{\mathrm{d}t} \\ &= \frac{\partial}{\partial t}\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t) + \nabla \log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\cdot\Big(-\frac{\beta_t}{2}\mathbf{R}_t - \frac{\beta_t}{2}\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\Big).\end{aligned}$$

Compared with Eq. (A9), we find:

$$\frac{\mathrm{d}}{\mathrm{d}t}\log p^{\theta}_{\mathcal{D},t}(\mathbf{R}_t) = -\nabla\cdot\Big(-\frac{\beta_t}{2}\mathbf{R}_t - \frac{\beta_t}{2}\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\Big) = \frac{D}{2}\beta_t + \frac{\beta_t}{2}\nabla\cdot\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t).$$

By integration w.r.t t, this gives:

$$\log p^{\theta}_{\mathcal{D},0}(\mathbf{R}_0) = \log p^{\theta}_{\tau}(\mathbf{R}_\tau) - \int_0^\tau \frac{\beta_t}{2}\nabla\cdot\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)\,\mathrm{d}t - \frac{D}{2}\int_0^\tau \beta_t\,\mathrm{d}t,$$

which gives Eq. (6).

The equivalent deterministic process described by Eq. (7) is called "probabilistic flow ODE" in machine learning literature [35]. Since this deterministic process produces the same marginal distribution $p^{\theta}_{\mathcal{D},t}$ (particularly the equilibrium distribution $p^{\theta}_{\mathcal{D},t}$), it can also be used to generate samples. Due to the deterministic nature, this approach enables more techniques that could accelerate the sampling process (Supplementary Sec. A.6).

A.2 Protein Conformation Sampling

A.2.1 Diffusion Process on Coarse-Grained Representation of Protein

Following the practice of successful protein structure prediction methods, e.g., AlphaFold [1], we use the coarse-grained representation for protein as the $\mathbf{R}$ variable. With residues treated as rigid bodies, proteins are represented by the coordinates $\mathbf{C}$ in $\mathbb{R}^3$ of alpha-carbon atoms and the orientations $\mathbf{Q}$ in the 3-dimensional rotation group (a.k.a special orthogonal group) SO(3) of all the residues. Following AlphaFold [1] (Supplementary 1.8.1), the coordinates and orientations are constructed using backbone atom positions from the experimental structure, followed by a Gram–Schmidt process.

For the coordinates $\mathbf{C}$, the standard diffusion modeling can be applied. However, it is not straightforward for the orientation, as SO(3) is a non-Euclidean manifold. Therefore, the forward and reverse diffusion processes need to be generalized. For this treatment, we adopted the technique from [7, 72].

Diffusion Process on the Orientations in the Special Orthogonal Group

Noting that SO(3) is a Lie group (i.e., a manifold that is also an algebraic group), we can represent its elements in its Lie algebra (i.e., the tangent space at the identity element) $\mathfrak{so}(3)$, which is a 3-dimensional linear space where vector addition and scaling are valid and random sampling is conventional. Specifically, a 3-dimensional vector $\mathbf{q} = (x, y, z) \in \mathfrak{so}(3)$ can be interpreted as defining the rotation axis and the rotation angle (the amount of rotation) of the corresponding rotation transformation on $\mathbb{R}^3$ by the direction and the norm of $\mathbf{q}$ as a usual vector in $\mathbb{R}^3$. The rotation matrix, as a form to represent an element in SO(3), can be constructed by:

$$\mathbf{Q} = \mathrm{Exp}\begin{pmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{pmatrix} \in \mathrm{SO}(3),\ \mathbf{q} = (x, y, z) \in \mathfrak{so}(3), \tag{A10}$$

in the conventional sense of a matrix exponential map.

To ease the calculation on SO(3), the forward diffusion process on it is taken as the corresponding Brownian motion (i.e., no drift term),

$$\mathrm{d}\mathbf{Q}_t = \sqrt{\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}}\,\mathrm{d}\tilde{\mathbf{B}}_t, \tag{A11}$$

where $\tilde{\mathbf{B}}_t$ denotes the Brownian motion on SO(3), and $\sqrt{\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}}$ (with σ_t strictly increasing) is a time-dilation factor. This process converges to the uniform distribution (maximal entropy distribution on a compact space; SO(3) is compact) as the corresponding p_{simple}. Simulation of the Brownian motion, say, from time step t_{i-1} to t_i, can be analogously done (up to $o(t_i - t_{i-1})$ discretization error) by adding a noise variable from the isotropic Gaussian distribution on SO(3) with variance $\sigma_{t_i}^2 - \sigma_{t_{i-1}}^2$, denoted as $\mathcal{IG}_{\mathrm{SO}(3)}(\mathbf{0}, \sigma_{t_i}^2 - \sigma_{t_{i-1}}^2)$. Drawing samples from $\mathcal{IG}_{\mathrm{SO}(3)}(\mathbf{0}, \sigma^2)$ can be done in $\mathfrak{so}(3)$ by uniformly sampling a direction in $\mathbb{R}^3$ for $\mathbf{q}$, and sampling the length of $\mathbf{q}$ (the length is within $[0, \pi]$) from the 1-dimensional distribution with the following density function:

$$\begin{aligned} p_{\mathcal{IG},\sigma^2}(\|\mathbf{q}\|) &= \frac{1 - \cos\|\mathbf{q}\|}{\pi}\tilde{p}_{\mathcal{IG},\sigma^2}(\|\mathbf{q}\|), \\ \text{where } \tilde{p}_{\mathcal{IG},\sigma^2}(\|\mathbf{q}\|) &:= \sum_{l=0}^{\infty}(2l+1)\mathrm{e}^{-l(l+1)(\sigma_{t_i}^2 - \sigma_{t_{i-1}}^2)}\frac{\sin((l+\frac{1}{2})\|\mathbf{q}\|)}{\sin(\frac{1}{2}\|\mathbf{q}\|)}. \end{aligned} \tag{A12}$$

The density function written in $\mathbf{q}$ under the Lebesgue measure in $\mathfrak{so}(3)$ is then $\mathcal{IG}_{\mathrm{SO}(3)}(\mathbf{q} \mid \mathbf{0}, \sigma^2) \propto p_{\mathcal{IG},\sigma^2}(\|\mathbf{q}\|)$.

We learn the score model (instead of the noise-predicting model) for this setting. As a gradient, the output of the score at time t becomes an element in the tangent space at $\mathbf{Q}_t$. Again thanks to the group structure, they can be mapped to the tangent space at the identity element, i.e. $\mathfrak{so}(3)$. The norm in

$\mathfrak{so}(3)$ consistent with the metric on SO(3) (i.e., the amount of rotation) is just the Euclidean 2-norm on the vector form $\mathbf{q}$. Hence in the Lie algebra $\mathfrak{so}(3)$, metric-related objects are the common Euclidean ones, including norm, gradient and divergence, which are to be used in the FPE Eq. (A1) hence the corresponding PIDP loss Eq. (4) and data-based loss Eq. (5). Nevertheless, there is a subtlety regarding the measure. To make the score function consistent with the diffusion process via the FPE, the density function should be taken w.r.t the uniform distribution on SO(3), which does not project to the Lebesgue measure ("uniform distribution") in $\mathfrak{so}(3)$. Instead, the SO(3) uniform distribution has the density:

$$p_{\mathrm{Unif}}(\mathbf{q}) := \frac{1-\cos\|\mathbf{q}\|}{\pi},\ (\|\mathbf{q}\| \leqslant \pi) \tag{A13}$$

under the Lebesgue measure in $\mathfrak{so}(3)$. Note this is also what p_{simple} takes. So the required score function of a distribution should be $\nabla \log \frac{p(\mathbf{q})}{p_{\mathrm{Unif}}(\mathbf{q})}$, where $p(\mathbf{q})$ is the density function of the distribution under the Lebesgue measure in $\mathfrak{so}(3)$. In particular, the score function of the isotropic Gaussian on SO(3) is $\nabla_{\mathbf{q}} \log \tilde{p}_{\mathcal{IG},\sigma^2}(\|\mathbf{q}\|)$.

With these facts, we are ready to develop PIDP and data-based training for a diffusion model that involves the SO(3) space. Recall that the coarse-grained representation $\mathbf{R}$ for proteins comprises alpha-carbon coordinates $\mathbf{C}$ and the orientations $\mathbf{Q}$ of residues. The orientations can be equivalently represented in $\mathfrak{so}(3)$ as $\mathbf{q}$, so we have $\mathbf{R} = (\mathbf{C}, \mathbf{q})$. Hence, the score model and the energy gradient (appearing in PIDP) take both $\mathbf{C}$ and $\mathbf{q}$ as input, and output vectors for both $\mathbf{C}$ and $\mathbf{q}$. Since the output vectors are in different spaces and have different losses, we split the output: $\mathbf{s}^{\theta}_{\mathcal{D},t} = (\mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},t}, \mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t})$, and $\nabla E_{\mathcal{D}} = (\nabla_{\mathbf{C}} E_{\mathcal{D}}, \nabla_{\mathbf{q}} E_{\mathcal{D}})$.

Now consider the PIDP loss for $\mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t}$. Following the forward process in Eq. (A11) and adopting the above definition of score function, the FPE Eq. (A1) leads to the loss:

$$\begin{aligned}&\left\| \frac{1}{2}\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}\Big(\nabla\big(\nabla\cdot \mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t}(\mathbf{C}_t,\mathbf{q}_t)\big) + \nabla\Big\|\mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t}(\mathbf{C}_t,\mathbf{q}_t)\Big\|^2\Big) - \frac{\partial}{\partial t}\mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t}(\mathbf{C}_t,\mathbf{q}_t)\right\|^2 \\ &+\lambda\left\|\mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},0}(\mathbf{C}_0,\mathbf{q}_0) + \nabla_{\mathbf{q}_0} E_{\mathcal{D}}(\mathbf{C}_0,\mathbf{q}_0)/(k_{\mathrm{B}}T)\right\|^2.\end{aligned} \tag{A14}$$

Following the pattern to run a PIDP loss as introduced in Sec. 2, the sample of $(\mathbf{C}_0, \mathbf{q}_0)$ is drawn from relevantly low-energy structures $\{(\mathbf{C}^{(m)}_{\mathcal{D},0}, \mathbf{q}^{(m)}_{\mathcal{D},0})\}_{m=1}^{M}$ for protein $\mathcal{D}$ (not necessarily following the equilibrium distribution), and the corresponding $(\mathbf{C}_t, \mathbf{q}_t)$ is sampled by letting $(\mathbf{C}_0, \mathbf{q}_0)$ undergo the forward process. We construct the forward process for $\mathbf{C}_t$ and $\mathbf{q}_t$ independently, so the marginal transition kernel can be decomposed as:

$$q(\mathbf{C}_t, \mathbf{q}_t \mid \mathbf{C}_0, \mathbf{q}_0) = q(\mathbf{C}_t \mid \mathbf{C}_0)\, q(\mathbf{q}_t \mid \mathbf{q}_0). \tag{A15}$$

Note that in the intermediate marginal distribution $q_t(\mathbf{C}_t, \mathbf{q}_t)$, the two variables are not independent since they are not in the equilibrium distribution. Hence, both the $\mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},t}$ model and the $\mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t}$ model take both $\mathbf{C}_t$ and $\mathbf{q}_t$ into their input. For $\mathbf{q}_t$ in Eq. (A15), recall that it is led by the Brownian motion on SO(3), whose marginal transition kernel is available in closed form:

$$q(\mathbf{q}_t \mid \mathbf{q}_0) = \mathcal{IG}_{\mathrm{SO}(3)}(\mathbf{q}_t \mid \mathbf{q}_0, \sigma_t^2), \tag{A16}$$

which turns sampling $\mathbf{q}_t$ straightforward following the above description to sample an $\mathcal{IG}_{\mathrm{SO}(3)}$. For $\mathbf{C}_t$ in Eq. (A15), it is sampled using Eq. (A20) detailed in the next part.

Data-based loss for $\mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t}$ is still based on the denoising score-matching loss Eq. (A3). The score function to be matched can be simplified as $\nabla_{\mathbf{q}_t} \log q(\mathbf{C}_t, \mathbf{q}_t \mid \mathbf{C}_0, \mathbf{q}_0) = \nabla_{\mathbf{q}_t} \log q(\mathbf{q}_t \mid \mathbf{q}_0)$ from Eq. (A15). This $q(\mathbf{q}_t \mid \mathbf{q}_0)$ is an $\mathcal{IG}_{\mathrm{SO}(3)}$ from Eq. (A16). Recalling the score function of $\mathcal{IG}_{\mathrm{SO}(3)}$, the data-based loss can then be written as:

$$\left\| \mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t}(\mathbf{C}_t, \mathbf{q}_t) - \nabla_{\mathbf{q}_t} \log \tilde{p}_{\mathcal{IG},\sigma_t^2}(\|\mathbf{q}_t\|) \right\|^2. \tag{A17}$$

The function $\tilde{p}_{\mathcal{IG},\sigma^2}(\|\mathbf{q}\|)$ is introduced in Eq. (A12). The sample $(\mathbf{C}_t, \mathbf{q}_t)$ for evaluating this loss is drawn following Eq. (A15), which again amounts to drawing $\mathbf{q}_t$ following Eq. (A16) and $\mathbf{C}_t$ following Eq. (A20) below. The required $(\mathbf{C}_0, \mathbf{q}_0)$ sample is drawn from the dataset $\{(\mathbf{C}^{(n)}_{\mathcal{D},0}, \mathbf{q}^{(n)}_{\mathcal{D},0})\}_{n=1}^{N_{\mathrm{data}}}$ that follows the equilibrium distribution of system $\mathcal{D}$.

Diffusion Process on the Alpha-Carbon Coordinates

To match the diffusion choice for SO(3) in Eq. (A11), we also adopt the Brownian motion as the forward process in the Euclidean space for the alpha-carbon coordinates:

$$\mathrm{d}\mathbf{C}_t = \sqrt{\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}}\, \mathrm{d}\mathbf{B}_t. \tag{A18}$$

This coincides with the choice in noise-conditioned score network [35, 73].

The PIDP loss for $\mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},t}$ from the FPE Eq. (A1) then becomes:

$$\begin{aligned} &\left\| \frac{1}{2} \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} \left(\nabla \left(\nabla \cdot \mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},t}(\mathbf{C}_t, \mathbf{q}_t) \right) + \nabla \left\| \mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},t}(\mathbf{C}_t, \mathbf{q}_t) \right\|^2 \right) - \frac{\partial}{\partial t} \mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},t}(\mathbf{C}_t, \mathbf{q}_t) \right\|^2 \\ &+ \lambda \left\| \mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},0}(\mathbf{C}_0, \mathbf{q}_0) + \nabla_{\mathbf{C}_0} E_{\mathcal{D}}(\mathbf{C}_0, \mathbf{q}_0)/(k_{\mathrm{B}}T) \right\|^2. \end{aligned} \tag{A19}$$

The sample of $(\mathbf{C}_0, \mathbf{q}_0)$ to evaluate the loss is again drawn from relevant structures $\{(\mathbf{C}^{(m)}_{\mathcal{D},0}, \mathbf{q}^{(m)}_{\mathcal{D},0})\}_{m=1}^{M}$ for protein $\mathcal{D}$, and the sample of $\mathbf{q}_t$ following

Eq. (A16). For the sample of $\mathbf{C}_t$, it is drawn from the marginal transition kernel of the diffusion process in Eq. (A18), which is a Gaussian distribution thus easy to draw:

$$q(\mathbf{C}_t \mid \mathbf{C}_0) = \mathcal{N}(\mathbf{C}_t \mid \mathbf{C}_0, \sigma_t^2 \mathbf{I}). \tag{A20}$$

Data-based loss for $\mathbf{s}_{\mathcal{D},t}^{(\mathbf{C}),\theta}$ also follows Eq. (A3), where the required score function is $\nabla_{\mathbf{C}_t} \log q(\mathbf{C}_t, \mathbf{q}_t \mid \mathbf{C}_0, \mathbf{q}_0) = \nabla_{\mathbf{C}_t} \log q(\mathbf{C}_t \mid \mathbf{C}_0)$ due to Eq. (A15), which leads to:

$$\left\| \mathbf{s}_{\mathcal{D},t}^{(\mathbf{C}),\theta}(\mathbf{C}_t, \mathbf{q}_t) - \nabla_{\mathbf{C}_t} \log q(\mathbf{C}_t \mid \mathbf{C}_0) \right\|^2. \tag{A21}$$

The sample of $(\mathbf{C}_0, \mathbf{q}_0)$ here is drawn from the dataset $\{(\mathbf{C}_{\mathcal{D},0}^{(n)}, \mathbf{q}_{\mathcal{D},0}^{(n)})\}_{n=1}^{N_{\text{data}}}$ that follows the equilibrium distribution, and sample of $(\mathbf{C}_t, \mathbf{q}_t)$ is drawn following Eqs. (A16, A20) . If substituting Eq. (A20) and expressing the loss in terms of the standard Gaussian sample $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ following the style of Eq. (5), then Eq. (A21) becomes: $\left\| \mathbf{s}_{\mathcal{D},t}^{(\mathbf{C}),\theta}(\mathbf{C}_0 + \sigma_t \epsilon_t, \mathbf{q}_t) + \frac{\epsilon_t}{\sigma_t} \right\|^2$. Nevertheless, such a reformulation does not easily apply to substitute $\mathbf{q}_t$ due to the complexity of $\mathcal{IG}_{\text{SO}(3)}$ in Eq. (A16).

Structure Sampling Using DiG

To generate structure samples using DiG, we find rather than directly simulating the reverse SDE analogous to Eq. (3), it is better to simulate the equivalent deterministic process defined by an ODE analogous to Eq. (7). The rationale of the equivalent ODE is explained in Supplementary Sec. A.1.3. Following the deduction there, the equivalent ODE for the diffusion processes in Eqs. (A11, A18) can be derived as:

$$\mathrm{d}\begin{pmatrix} \mathbf{C}_t \\ \mathbf{q}_t \end{pmatrix} = -\frac{1}{2}\frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t}\begin{pmatrix} \mathbf{s}_{\mathcal{D},t}^{(\mathbf{C}),\theta}(\mathbf{C}_t, \mathbf{q}_t) \\ \mathbf{s}_{\mathcal{D},t}^{(\mathbf{q}),\theta}(\mathbf{C}_t, \mathbf{q}_t) \end{pmatrix}\mathrm{d}t = -\frac{1}{2}\begin{pmatrix} \mathbf{s}_{\mathcal{D},t}^{(\mathbf{C}),\theta}(\mathbf{C}_t, \mathbf{q}_t) \\ \mathbf{s}_{\mathcal{D},t}^{(\mathbf{q}),\theta}(\mathbf{C}_t, \mathbf{q}_t) \end{pmatrix}\mathrm{d}\sigma_t^2. \tag{A22}$$

The simulation on $\mathbf{C}_t$ is thus:

$$\mathbf{C}_{t_{i-1}} = \mathbf{C}_{t_i} + \frac{1}{2}(\sigma_{t_i}^2 - \sigma_{t_{i-1}}^2)\mathbf{s}_{\mathcal{D},t_i}^{(\mathbf{C}),\theta}(\mathbf{C}_{t_i}, \mathbf{q}_{t_i}). \tag{A23}$$

The simulation on $\mathbf{q}_t$ can be done similarly by $\mathbf{q}_{t_{i-1}} = \mathbf{q}_{t_i} + \frac{1}{2}(\sigma_{t_i}^2 - \sigma_{t_{i-1}}^2)\mathbf{s}_{\mathcal{D},t_i}^{(\mathbf{q}),\theta}(\mathbf{C}_{t_i}, \mathbf{q}_{t_i})$. This discretization on $\mathfrak{so}(3)$ is equivalent to discretization on SO(3) in the sense that their one-step difference is $o(t_i - t_{i-1})$, but the simulation in $\mathfrak{so}(3)$ directly faces the risk that the discretization error may lead the $\mathbf{q}$ variable going out of the domain (i.e., $\|\mathbf{q}\| > \pi$), as there is no mechanism to guarantee the constraint. We therefore carry out the simulation

in SO(3) instead:

$$\mathbf{Q}_{t_{i-1}} = \mathrm{Exp}\left(\frac{1}{2}(\sigma_{t_i}^2 - \sigma_{t_{i-1}}^2)\mathbf{s}_{\mathcal{D},t_i}^{(\mathbf{Q}),\theta}(\mathbf{C}_{t_i}, \mathbf{q}_{t_i})\right)\mathbf{Q}_{t_i}, \tag{A24}$$

where Exp is the conventional matrix exponent, and $\mathbf{s}_{\mathcal{D},t_i}^{(\mathbf{Q}),\theta}$ denotes the skew-symmetric matrix by organizing the outputs of $\mathbf{s}_{\mathcal{D},t_i}^{(\mathbf{q}),\theta}$ in the same way as converting $\mathbf{Q}$ and $\mathbf{q}$ Eq. (A10). Note that the matrix exponent of a skew-symmetric matrix is a rotation matrix, Eq. (A24) then guarantees $\mathbf{Q}_{t_{i-1}} \in \mathrm{SO}(3)$ whenever $\mathbf{Q}_{t_i} \in \mathrm{SO}(3)$. Alg. 5 summarizes the sampling procedure.

Interpolation between Protein States Using DiG

The deterministic nature of the ODE Eq. (A22) establishes a deterministic map between a real state $\mathbf{R}_0$ and the corresponding latent state $\mathbf{R}_\tau$. This enables the complicated interpolation between two given states, $\mathbf{R}_0^{(A)} = (\mathbf{C}_0^{(A)}, \mathbf{q}_0^{(A)})$ and $\mathbf{R}_0^{(B)} = (\mathbf{C}_0^{(B)}, \mathbf{q}_0^{(B)})$, by a simpler interpolation in the latent space of $\mathbf{R}_\tau$ where the distribution is simple. For the alpha-carbon coordinates $\mathbf{C}_0^{(A)}$ and $\mathbf{C}_0^{(B)}$, we apply linear interpolation to their corresponding latent states $\mathbf{C}_\tau^{(A)}$ and $\mathbf{C}_\tau^{(B)}$ through the ODE, and then transform the line to the real-state space of $\mathbf{C}_0$ by the ODE reversely. Since the coordinate distribution in the latent space is standard Gaussian, which has a convex contour, linear interpolation there would pass through high-probability regions. For the residue orientations $\mathbf{q}_0^{(A)}$ and $\mathbf{q}_0^{(B)}$, as the corresponding latent states $\mathbf{q}_\tau^{(A)}$ and $\mathbf{q}_\tau^{(B)}$ lie in the product space of $\mathfrak{so}(3)$ which is non-Euclidean, we leverage spherical linear interpolation which gives the geodesic in $\mathfrak{so}(3)$ between two given end states, in place of the linear interpolation which is the geodesic in the Euclidean space. Explicitly, the interpolation curves in the latent space are:

$$\mathbf{C}_\tau^{(\eta)} = (1-\eta)\mathbf{C}_\tau^{(A)} + \eta\mathbf{C}_\tau^{(B)}, \tag{A25}$$

$$\mathbf{q}_\tau^{(\eta)} = (\mathbf{q}_\tau^{(B)}(\mathbf{q}_\tau^{(A)})^{-1})^\eta \mathbf{q}_\tau^{(A)}, \tag{A26}$$

where $\eta \in [0,1]$ parameterizes the interpolation curve. Subsequently, $(\mathbf{C}_\tau^{(\eta)}, \mathbf{q}_\tau^{(\eta)})$ in Eqs. (A25, A26) are taken as the starting state to simulate the ODEs Eqs. (A23, A24) reversely to generate the interpolated structures at the parameter η along the transition pathway between state A and B.

A.2.2 Model Specification

Following the practice of AlphaFold [1], we use amino acid sequences as the molecular descriptor $\mathcal{D}$ for proteins. To process the sequence to generate informative abstract representations for the feed into DiG, we follow the data processing method in the training stage of AlphaFold and leverage the pre-trained Evoformer to produce node and pair representations. Conditioned on representations of proteins, DiG aims to gradually transform random noise

to reasonable protein structures following the equilibrium distribution. Considering that protein simulation trajectories that are long enough and reach equilibrium distribution are very rare, for protein conformation sampling, as in Sec. 2, we pre-train DiG with PIDP first, and then further improved the performance by training the model with a small amount of simulation data.

Algorithm 1 Protein score model PIDP training (single step)

Require: Score model $\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R})$ to be trained, boundary loss weight λ, randomly sampled system $\mathcal{D}$, full-atom energy function $E_{\mathcal{D}}$ for system $\mathcal{D}$, a randomly sampled relevant full-atom protein structure $\bar{\mathbf{R}}^{(m)}_{\mathcal{D},0}$ for system $\mathcal{D}$, randomly sampled time step $t \in [0, \tau]$.

1: Construct $\mathbf{R}^{(m)}_{\mathcal{D},0} := (\mathbf{C}^{(m)}_{\mathcal{D},0}, \mathbf{q}^{(m)}_{\mathcal{D},0})$ from the sampled $\bar{\mathbf{R}}^{(m)}_{\mathcal{D},0}$;
2: Compute the energy gradient $\nabla_{\mathbf{C}} E_{\mathcal{D}}(\mathbf{C}^{(m)}_{\mathcal{D},0}, \mathbf{q}^{(m)}_{\mathcal{D},0})$ and $\nabla_{\mathbf{q}} E_{\mathcal{D}}(\mathbf{C}^{(m)}_{\mathcal{D},0}, \mathbf{q}^{(m)}_{\mathcal{D},0})$ for the alpha-carbon coordinates and residue orientations, respectively, from the full-atom energy function $E_{\mathcal{D}}$ using Alg. 2;
3: Sample $(\mathbf{C}^{(m)}_{\mathcal{D},t}, \mathbf{q}^{(m)}_{\mathcal{D},t})$ from $(\mathbf{C}^{(m)}_{\mathcal{D},0}, \mathbf{q}^{(m)}_{\mathcal{D},0})$ using Eq. (A20) and Eq. (A16);
4: Evaluate the PIDP loss $\mathcal{L}_{\text{PIDP}}$ as Eq. (A14) + Eq. (A19);
5: Update the model parameter θ by performing an optimization step on $\mathcal{L}_{\text{PIDP}}$ with respect to θ.

Algorithm 2 Compute the energy gradient on orientations and coordinates

Require: Energy function E, full atom protein structure $\bar{\mathbf{R}}$ with I amino acid residues.

1: Construct $\mathbf{R} := (\mathbf{C}, \mathbf{Q})$ from $\bar{\mathbf{R}}$;
2: Compute the energy $E = E(\bar{\mathbf{R}})$;
3: **for** each residue i in $1, \cdots, I$ **do**
4: Set $\mathbf{X}_i$ as the coordinates of all N_i atoms in the residue i;
5: $\mathbf{X}_{i,\text{rel}} := (\mathbf{X}_i - \mathbf{C}_i)\mathbf{Q}_i^{\top}$ (c.f. Eq. (A27));
6: $\mathbf{g}_{\mathbf{C}_i} := \sum_{a=1}^{N_i} \nabla_{\mathbf{X}_{i,a}} E$ (c.f. Eq. (A28));
7: $\mathbf{g}_{\mathbf{q}_i} := \sum_{a=1}^{N_i} \mathbf{X}^{\top}_{i,\text{rel},a} \nabla_{\mathbf{q}_i} \mathbf{Q}_i \nabla_{\mathbf{X}_{i,a}} E$ (c.f. Eq. (A29));
8: **end for**
9: Return $\nabla_{\mathbf{C}} E := \{\mathbf{g}_{\mathbf{C}_i}\}_{i=1}^{I}$ and $\nabla_{\mathbf{q}} E := \{\mathbf{g}_{\mathbf{q}_i}\}_{i=1}^{I}$.

We first train DiG by minimizing a PIDP loss that aligns the score model with the gradients of the energy function and enforces the boundary conditions. Alg. 1 outlines the training process. The PIDP training requires the energy gradient label. This is facilitated by an energy function from OpenMM [74] at the full-atom level. But as we are adopting a coarse-grained representation for proteins, so the energy gradients w.r.t alpha carbon coordinates and residue

Algorithm 3 Estimate $\nabla(\nabla \cdot \mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t))$ by Hutchinson's trace estimator

Require: Score model $\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R})$, protein structure $\mathbf{R}_t$, number of random vectors N_{est}.

1: Sample N_{est} random vectors of the same dimension as $\mathbf{R}$: $\left\{\mathbf{v}^{(n)}\right\}_{n=1}^{N_{\text{est}}} \overset{\text{i.i.d.}}{\sim}$ Rademacher(0.5);

2: $\mathbf{g} := \frac{1}{N_{\text{est}}} \sum_{n=1}^{N_{\text{est}}} \nabla_{\mathbf{R}_t}(\mathbf{v}^{(n)\top} \nabla_{\mathbf{R}_t}(\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t)^{\top}\mathbf{v}^{(n)}))$;

3: Return $\mathbf{g}$ as an approximation to $\nabla(\nabla \cdot \mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t))$.

orientations are expected. For this, we leverage the rigid-body assumption and the chain rule to derive the conversion, as shown in Alg. 2.

Here we briefly explain the derivation in Alg. 2. The rigid-body assumption states that for each residue $\imath$ with $N_\imath$ atoms, the relative coordinates $\mathbf{X}_{\imath,\text{rel}} \in \mathbb{R}^{N_\imath \times 3}$ of all its atoms w.r.t its alpha-carbon at $\mathbf{C}_\imath \in \mathbb{R}^{1\times 3}$ in a standard coordinate system is fixed. Under this assumption, if the coarse-grained representation of this residue is $(\mathbf{C}_\imath, \mathbf{Q}_\imath)$ where $\mathbf{Q}_\imath$ is the orientation of the residue relative to the standard coordinate system, then the absolute coordinates of these atoms are:

$$\mathbf{X}_\imath = \mathbf{C}_\imath + \mathbf{X}_{\imath,\text{rel}}\mathbf{Q}_\imath, \tag{A27}$$

and if considering a protein with I residues, the full-atom coordinates are $\bar{\mathbf{R}} := \{\mathbf{C}_\imath + \mathbf{X}_{\imath,\text{rel}}\mathbf{Q}_\imath\}_{\imath=1}^{I}$. So in this way, we can convert the full-atom energy function $E(\bar{\mathbf{R}}) = E(\{\mathbf{X}_\imath\}_{\imath=1}^{I})$ as a function of the coarse-grained coordinates $\mathbf{R} = (\mathbf{C}, \mathbf{Q})$ using Eq. (A27). The gradient w.r.t $\mathbf{C}_\imath$ is then:

$$\mathbf{g}_{\mathbf{C}_\imath} := \nabla_{\mathbf{C}_\imath} E = (\nabla_{\mathbf{C}_\imath}\mathbf{X}_\imath)^{\top} \nabla_{\mathbf{X}_\imath} E = \sum_{a=1}^{N_\imath} \nabla_{\mathbf{X}_{\imath,a}} E, \tag{A28}$$

where $(\nabla_{\mathbf{C}_\imath}\mathbf{X}_\imath)_{a\mu,\nu} := \frac{\partial \mathbf{X}_{\imath,a,\mu}}{\partial \mathbf{C}_{\imath,\nu}}$ is the Jacobian matrix (here $\mu, \nu \in \{1,2,3\}$ indices the spacial dimension), and the last equality holds since the Jacobian is $(\nabla_{\mathbf{C}_\imath}\mathbf{X}_\imath)_{a\mu,\nu} = \delta_{\mu\nu}$ meaning that this matrix element is one if $\mu = \nu$ or it is zero.

As for the gradient w.r.t the orientation, the $\mathfrak{so}(3)$ representation denoted as $\mathbf{q}$ is finally required. The conversion from $\mathbf{q}$ and $\mathbf{Q}$ is given by Eq. (A10). Together with the rigid-body assumption Eq. (A27), the gradient is:

$$\begin{aligned}\mathbf{g}_{\mathbf{q}_\imath} &:= \nabla_{\mathbf{q}_\imath} E = (\nabla_{\mathbf{q}_\imath}\mathbf{X}_\imath)^{\top} \nabla_{\mathbf{X}_\imath} E = (\nabla_{\mathbf{Q}_\imath}\mathbf{X}_\imath \nabla_{\mathbf{q}_\imath}\mathbf{Q}_\imath)^{\top} \nabla_{\mathbf{X}_\imath} E \\ &= \sum_{a=1}^{N_\imath} \mathbf{X}^{\top}_{\imath,\text{rel},a} \nabla_{\mathbf{q}_\imath}\mathbf{Q}_\imath \nabla_{\mathbf{X}_{\imath,a}} E,\end{aligned} \tag{A29}$$

Algorithm 4 Protein score model data-based training (single step)

Require: Score model $\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R})$ to be trained, randomly sampled system $\mathcal{D}$, a randomly sampled coarse-grained protein structure $\mathbf{R}^{(n)}_{\mathcal{D},0} = (\mathbf{C}^{(n)}_{\mathcal{D},0}, \mathbf{q}^{(n)}_{\mathcal{D},0})$ from the MD simulation data for system $\mathcal{D}$, randomly sampled time step $t \in [0, \tau]$.

1: Sample $(\mathbf{C}^{(n)}_{\mathcal{D},t}, \mathbf{q}^{(n)}_{\mathcal{D},t})$ from $(\mathbf{C}^{(n)}_{\mathcal{D},0}, \mathbf{q}^{(n)}_{\mathcal{D},0})$ using Eq. (A20) and Eq. (A16);
2: Evaluate the data-based loss $\mathcal{L}_{\text{data}}$ as Eq. (A17) + Eq. (A21);
3: Update the model parameter θ by performing an optimization step on $\mathcal{L}_{\text{data}}$ with respect to θ.

where the last term means $(\mathbf{g}_{\mathbf{q}_\imath})_\gamma = \sum_{a=1}^{N_\imath} \mathbf{X}^\top_{\imath,\text{rel},a} \frac{\partial \mathbf{Q}_\imath}{\partial \mathbf{q}_{\imath,\gamma}} \nabla_{\mathbf{X}_{\imath,a}} E$, where $\gamma \in \{1,2,3\}$ indices one of the three dimensions of $\mathbf{q}_\imath$, and $\frac{\partial \mathbf{Q}_\imath}{\partial \mathbf{q}_{\imath,\gamma}}$ is the 3×3 matrix composed of the partial derivatives from Eq. (A10). In the equation, again $\nabla_{\mathbf{q}_\imath}\mathbf{X}_\imath$, $\nabla_{\mathbf{Q}_\imath}\mathbf{X}_\imath$ and $\nabla_{\mathbf{q}_\imath}\mathbf{Q}_\imath$ are Jacobian matrices, and the second last equality holds due to the chain rule of differentiation. From the rigid-body assumption Eq. (A27), $(\nabla_{\mathbf{Q}_\imath}\mathbf{X}_\imath)_{a\nu',\mu\nu} = \frac{\partial \mathbf{X}_{\imath,a,\nu'}}{\partial \mathbf{Q}_{\imath,\mu,\nu}} = \mathbf{X}_{\imath,\text{rel},a,\mu}\delta_{\nu\nu'}$, so:

$$[(\nabla_{\mathbf{Q}_\imath}\mathbf{X}_\imath \nabla_{\mathbf{q}_\imath}\mathbf{Q}_\imath)^\top \nabla_{\mathbf{X}_\imath} E]_\gamma = \sum_{a,\nu',\mu,\nu} (\nabla_{\mathbf{Q}_\imath}\mathbf{X}_\imath)_{a\nu',\mu\nu} (\nabla_{\mathbf{q}_\imath}\mathbf{Q}_\imath)_{\mu\nu,\gamma} (\nabla_{\mathbf{X}_\imath} E)_{a\nu'}$$

$$= \sum_{a,\mu,\nu} \mathbf{X}_{\imath,\text{rel},a,\mu} (\nabla_{\mathbf{q}_\imath}\mathbf{Q}_\imath)_{\mu\nu,\gamma} (\nabla_{\mathbf{X}_\imath} E)_{a\nu} = \sum_a \mathbf{X}^\top_{\imath,\text{rel},a} \frac{\partial \mathbf{Q}_\imath}{\partial \mathbf{q}_{\imath,\gamma}} \nabla_{\mathbf{X}_{\imath,a}} E,$$

which gives the last equality.

Moreover, to avoid costly divergence evaluation, we use Hutchinson's trace estimator in Alg. 3 to handle $\nabla(\nabla \cdot \mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t))$ (recall that $\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R}_t) = (\mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},t}(\mathbf{C}_t, \mathbf{q}_t), \mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t}(\mathbf{C}_t, \mathbf{q}_t))$) in Eqs. (A14, A19) .

Optimizing PIDP from random initialization of the deep learning model is extremely hard due to the complex landscapes of the training objectives in Eqs. (A14, A19) . Therefore, a good initialization of the model and some training techniques are necessary to stabilize the optimization of PIDP. To this end, before performing PIDP, we train DiG on the experimental structures and use it as a more stable initialization for PIDP. See Supplementary Sec. C for more details.

Next, we pick about 1000 protein complexes in PDBbind database [75], and simulate them with GROMACS, together with about 200 proteins in GPCRmd [76] dataset to perform DiG training with simulation data (See more details in Supplementary Sec. D.1). We further train the score model pre-trained by PIDP by minimizing a score matching loss from Eq. (5) that directly supervises the score model with the empirical data distribution that approximates the equilibrium distribution, using protein structures $\mathbf{R}^{(n)}_{\mathcal{D},0}$ for each system $\mathcal{D}$. See Supplementary Sec. B and C for more details on the model and training.

Algorithm 5 Protein structure sampling

Require: A trained score model $\mathbf{s}^{\theta}_{\mathcal{D},t}(\mathbf{R})$, target protein system $\mathcal{D}$.

1: Initialize random structure $\mathbf{R}_\tau := (\mathbf{C}_\tau, \mathbf{q}_\tau)$, where $\mathbf{C}_\tau \sim \mathcal{N}(\mathbf{0}, \sigma_\tau^2 \mathbf{I})$, and $\mathbf{q}_\tau \sim p_{\text{Unif}}$ on $\mathfrak{so}(3)$ defined in Eq. (A13). $t_N := \tau$.
2: **for** i in $N, \cdots, 1$ **do**
3: $\quad t_{i-1} := \frac{i-1}{N}\tau$;
4: $\quad \mathbf{C}_{t_{i-1}} := \mathbf{C}_{t_i} + \frac{1}{2}(\sigma_{t_i}^2 - \sigma_{t_{i-1}}^2)\mathbf{s}^{(\mathbf{C}),\theta}_{\mathcal{D},t_i}(\mathbf{C}_{t_i}, \mathbf{q}_{t_i})$ (c.f. Eq. (A23));
5: $\quad$ Construct $\mathbf{Q}_{t_i}$ from $\mathbf{q}_{t_i}$, and $\mathbf{s}^{(\mathbf{Q}),\theta}_{\mathcal{D},t_i}(\mathbf{C}_{t_i}, \mathbf{q}_{t_i})$ from $\mathbf{s}^{(\mathbf{q}),\theta}_{\mathcal{D},t_i}(\mathbf{C}_{t_i}, \mathbf{q}_{t_i})$, using Eq. (A10);
6: $\quad \mathbf{Q}_{t_{i-1}} := \text{Exp}\left(\frac{1}{2}(\sigma_{t_i}^2 - \sigma_{t_{i-1}}^2)\mathbf{s}^{(\mathbf{Q}),\theta}_{\mathcal{D},t_i}(\mathbf{C}_{t_i}, \mathbf{q}_{t_i})\right)\mathbf{Q}_{t_i}$ (c.f. Eq. (A24));
7: **end for**
8: Return the sampled structure $\mathbf{R}_0 := (\mathbf{C}_0, \mathbf{q}_0)$.

After training, amino acid sequences serve as input descriptors, denoted as $\mathcal{D}$, for sampling protein conformations via DiG. During the sampling procedure, an initial random structure is generated, which subsequently is transformed into a physically plausible conformation, as shown in Alg. 5.

A.3 Ligand Structure Sampling around Binding Sites

In contrast to the coarse-grained representation employed in protein conformation sampling, we train DiG of ligand structure sampling with all-atom (except hydrogens) representations, which offer a more precise description of atomic interactions between the binding site (pocket) and ligand during the sampling of ligand-binding structures with proteins. However, the time complexity and memory usage associated with attention layers in Transformer-based architectures exhibit a quadratic increase with respect to the number of input nodes. This becomes impractical when the atom count surpasses one thousand. Consequently, we restrict our model to incorporate only the atoms of the ligand and the protein atoms in close proximity to the pocket, using a distance threshold. This threshold is set to 10 Å for the side length.

DiG defines a distribution over the vector space $\bar{\mathbf{R}} := (\bar{\mathbf{R}}_{\text{Rec}}, \bar{\mathbf{R}}_{\text{Lig}})$, where $\bar{\mathbf{R}}_{\text{Rec}}$ and $\bar{\mathbf{R}}_{\text{Lig}}$ are the absolute coordinates of the near-site receptor and ligand atoms, respectively. Since the receptor atom coordinates may have different distribution centers for different proteins while the diffusion process always starts from a zero-centered Gaussian, we use the near-site receptor atom coordinates $\bar{\mathbf{R}}^{\star}_{\text{Rec}}$ from the crystal structure of the protein in the PDBBind database [75] as a reference structure, and let the model predict the residue. This effectively shifts the diffusion process to $\mathbf{R} := (\bar{\mathbf{R}}_{\text{Rec}} - \bar{\mathbf{R}}^{\star}_{\text{Rec}}, \bar{\mathbf{R}}_{\text{Lig}})$. DiG then generates the binding structures for the ligand and the protein pocket by reversing the diffusion process, as shown in Eq. (3).

To train DiG, we reuse the simulation data of protein complexes in PDBbind, as detailed in Supplementary Sec. C. The data preprocessing and featurization follows [77, 78]. The atom representations are further embedded

into real-valued embedding vectors for a Graphormer [2]. In this context, performing PIDP is not feasible due to limitations imposed by the energy function. Specifically, DiG only considers atoms surrounding the binding site, while conventional force fields necessitate the inclusion of all atoms. As such, we sought to enhance ligand sampling performance within the pocket by conducting a pre-training task focused on binding structure prediction, utilizing the CrossDocked dataset [79]. Further information regarding model architecture and training can be found in Supplementary Sec. B and C.

A.4 Catalyst-Adsorbate Sampling

For catalyst-adsorbate sampling, DiG adopts the same input representation strategy employed in the OC20 dataset [10], employing the descriptor $\mathcal{D}$ to characterize the system. Specifically, besides the atom types $\mathcal{Z}$, also provided from the OC20 dataset are the absolute coordinates $\bar{\mathbf{R}}^{\star}_{\text{base}}$ for non-surface catalyst atoms, $\bar{\mathbf{R}}^{\star}_{\text{Cat}}$ for surface catalyst atoms, and the initial absolute coordinates $\bar{\mathbf{R}}^{\star}_{\text{Ad}}$ for adsorbate atoms prior to relaxation. Consequently, the system descriptor for this task was defined as $\mathcal{D} := (\mathcal{Z}, \bar{\mathbf{R}}^{\star}_{\text{base}}, \bar{\mathbf{R}}^{\star}_{\text{Cat}}, \bar{\mathbf{R}}^{\star}_{\text{Ad}})$. The microscopic state of the system $\bar{\mathbf{R}} := (\bar{\mathbf{R}}_{\text{Cat}}, \bar{\mathbf{R}}_{\text{Ad}})$ encompasses the absolute coordinates $\bar{\mathbf{R}}_{\text{Cat}}$ of surface catalyst atoms and $\bar{\mathbf{R}}_{\text{Ad}}$ of the adsorbate atoms. Similar to the ligand-receptor sampling case, to ease the prediction of different distribution centers for different systems, we leverage the absolute coordinates in the descriptor to define the diffusion-process variable as relative coordinates, i.e., $\mathbf{R} := (\mathbf{R}_{\text{Cat}}, \mathbf{R}_{\text{Ad}})$ where $\mathbf{R}_{\text{Cat}} := \bar{\mathbf{R}}_{\text{Cat}} - \bar{\mathbf{R}}^{\star}_{\text{Cat}}$ and $\mathbf{R}_{\text{Ad}} := \bar{\mathbf{R}}_{\text{Ad}} - \bar{\mathbf{R}}^{\star}_{\text{Ad}}$, whose distribution center is largely aligned across different systems.

During the reverse diffusion process, including the initial structure $\bar{\mathbf{R}}^{\star}_{\text{Ad}}$ into the model input is found crucial for stable training. In the Graphormer model, in addition to the diffusion-variable $\mathbf{R}$ in the input, the initial structure $\bar{\mathbf{R}}^{\star}_{\text{Ad}}$ is also encoded as an additional structural attention bias term, as in [80]. More specifically, the pairwise distance between atoms in $\bar{\mathbf{R}}^{\star}_{\text{Ad}}$ was calculated, which is then encoded into a K-dimensional feature using K radial basis function (RBF) kernels with learnable means and variances. Likewise, the initial positions of the atoms are encoded as extra node features and incorporated into the node representation. By summing the pairwise features and aggregating them with the node features, per-atom features are updated and projected into the atom embedding dimension within the model. Further details regarding the structural attention bias can be found in Supplementary Sec. B.3.

Catalyst systems in OC20 are inorganic and lack bond information between atoms. However, adsorbates are organic, and the bonds between atoms in adsorbates can benefit the model in generating more physically accurate adsorbate structures. Consequently, we explicitly incorporated the 2D topology of adsorbates, featuring bonds connecting the atoms, within the model. Bonds are generated using the initial structure of the adsorbate and encoded in the same manner as Graphormer [2]. The spatial encoding, centrality encoding, and edge encoding of Graphormer are utilized alongside the encodings derived from 3D information. It is important to note that, in some instances, bonds

in adsorbates may break upon adsorption to the catalyst surface. Explicitly encoding bond information does not imply the enforcement of bonded atoms to remain close in the sampled structures. Instead, the model is allowed to learn when to separate two bonded atoms.

The training and sampling of DiG adhere to the general description in the main text (Sec. 2). The training loss is based on Eq. (A6), and Alg. 6 shows the detailed training process. For sampling new structures using DiG, in accordance with the training process, it is also conducted on the relative coordinates w.r.t the initial structure. Alg. 7 outlines the sampling process.

Algorithm 6 Catalyst-adsorbate score model training (single step)

Require: Noise-predicting model $\epsilon^{\theta}_{\mathcal{D},t}(\mathbf{R})$ to be trained, randomly sampled time step $t \in [0, \tau]$, randomly sampled system with descriptor $\mathcal{D} = (\mathcal{Z}, \bar{\mathbf{R}}^{\star}_{\text{base}}, \bar{\mathbf{R}}^{\star}_{\text{Cat}}, \bar{\mathbf{R}}^{\star}_{\text{Ad}})$, let $\bar{\mathbf{R}}^{\star} := \left(\bar{\mathbf{R}}^{\star}_{\text{Cat}}, \bar{\mathbf{R}}^{\star}_{\text{Ad}}\right)$, randomly sampled catalyst-adsorbent structure $\bar{\mathbf{R}}^{(n)}_{\mathcal{D},0} = (\bar{\mathbf{R}}^{(n)}_{\mathcal{D},\text{Cat},0}, \bar{\mathbf{R}}^{(n)}_{\mathcal{D},\text{Ad},0})$ from the MD simulation data for this system $\mathcal{D}$.

1: Let $\mathbf{R}^{(n)}_{\mathcal{D},0} := \bar{\mathbf{R}}^{(n)}_{\mathcal{D},0} - \bar{\mathbf{R}}^{\star}$;
2: Sample noise variable $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ in the same dimension as $\mathbf{R}^{(n)}_{\mathcal{D},0}$;
3: $\mathbf{R}^{(n)}_{\mathcal{D},t} := \alpha_t \mathbf{R}^{(n)}_{\mathcal{D},0} + \sqrt{1-\alpha_t^2}\epsilon_t$ (c.f. Eq. (A4));
4: Evaluate the loss $\left\| \epsilon^{\theta}_{\mathcal{D},t}(\mathbf{R}^{(n)}_{\mathcal{D},t} + \bar{\mathbf{R}}^{\star}) - \epsilon_t \right\|^2$ (c.f. Eq. (A6));
5: Update the model parameter θ by performing an optimization step on the loss with respect to θ.

Algorithm 7 Catalyst-adsorbate structure sampling

Require: A trained noise-predicting model $\epsilon^{\theta}_{\mathcal{D},t}(\mathbf{R})$, the descriptor $\mathcal{D} = (\mathcal{Z}, \bar{\mathbf{R}}^{\star}_{\text{base}}, \bar{\mathbf{R}}^{\star}_{\text{Cat}}, \bar{\mathbf{R}}^{\star}_{\text{Ad}})$ of the target system, let $\bar{\mathbf{R}}^{\star} := \left(\bar{\mathbf{R}}^{\star}_{\text{Cat}}, \bar{\mathbf{R}}^{\star}_{\text{Ad}}\right)$.

1: Sample a noisy structure $\mathbf{R}_{\tau} \sim p_{\text{simple}} = \mathcal{N}(\mathbf{0}, \mathbf{I})$;
2: **for** i in $N, \cdots, 1$ **do**
3: $\quad t_{i-1} := \frac{i-1}{N}\tau$;
4: $\quad$ Let $\beta_i := \frac{\tau}{N}\beta_{t_i}$, $\alpha_i := \prod_{j=1}^{i}\sqrt{1-\beta_j}$;
5: $\quad$ Sample a noise variable $\epsilon_{t_i} \sim \mathcal{N}(\mathbf{0}, \beta_i \mathbf{I})$ in the same dimension as $\mathbf{R}_{t_i}$;
6: $\quad \mathbf{R}_{t_{i-1}} := \frac{1}{\sqrt{1-\beta_i}}\left(\mathbf{R}_{t_i} - \frac{\beta_i}{\sqrt{1-\alpha_i^2}}\epsilon^{\theta}_{\mathcal{D},t_i}(\mathbf{R}_{t_i} + \bar{\mathbf{R}}^{\star})\right) + \epsilon_{t_i}$ (c.f. Eqs. (3, A5));
7: **end for**
8: **Return** $\mathbf{R}_0 + \bar{\mathbf{R}}^{\star}$.

As catalysts are periodic systems along the x and y directions, DiG expands the unit cell in these dimensions before feeding a system into the model. Providing an exact descriptor of the infinitely repeated system in the model is non-trivial; instead, DiG adopts a simple yet effective approach by establishing

a local cutoff for the infinitely repeating system. Specifically, an atom outside the unit cell is included in the model only if its distance to any atom inside the unit cell is within a threshold. A distance threshold of 6 Å is used in the experiments. In each layer of the transformer model, the representation of a repeated atom outside the unit cell is enforced to be identical to the representation of the corresponding atom in the unit cell. More details about handling periodic boundary conditions can be found in Supplementary Sec. B.5.

Lastly, to ensure a good initialization for the diffusion model, a model pretrained on the IS2RS task of OC20 is used to initialize the weights, except for the time step embedding, which is dedicated to the diffusion task.

A.5 Property-Guided Structure Generation

For modeling carbon polymorphs, the structural variable needs to represent the unit cell which defines a spatial period in the crystal, and the absolute coordinates $\bar{\mathbf{X}}$ of the carbon atoms in the unit cell. The unit cell is a parallelepiped to guarantee periodicity, so it can be determined by the coordinates of 4 non-coplanar vertices, say $\bar{\mathbf{c}}_0$, $\bar{\mathbf{c}}_0 + \bar{\mathbf{l}}_x$, $\bar{\mathbf{c}}_0 + \bar{\mathbf{l}}_y$, and $\bar{\mathbf{c}}_0 + \bar{\mathbf{l}}_z$, where $\bar{\mathbf{c}}_0$ is the origin of the unit cell, and $\bar{\mathbf{L}} := \{\bar{\mathbf{l}}_x, \bar{\mathbf{l}}_y, \mathbf{l}_z\}$ are known as the lattice vectors. (The locations of the other 4 vertices can be determined as $\bar{\mathbf{c}}_0 + \bar{\mathbf{l}}_x + \bar{\mathbf{l}}_y$, $\bar{\mathbf{c}}_0 + \bar{\mathbf{l}}_x + \bar{\mathbf{l}}_z$, $\bar{\mathbf{c}}_0 + \bar{\mathbf{l}}_y + \bar{\mathbf{l}}_z$, and $\bar{\mathbf{c}}_0 + \bar{\mathbf{l}}_x + \bar{\mathbf{l}}_y + \bar{\mathbf{l}}_z$.) In the structure representation, the origin $\bar{\mathbf{c}}_0$ is fixed as $\mathbf{0}$, so the unit cell is fully determined by the lattice vectors $\bar{\mathbf{L}}$.

Similar to the cases of protein-ligand sampling in Supplementary Sec. A.3 and catalyst-adsorbate sampling in Supplementary Sec. A.4, we take the diffusion-process variable $\mathbf{R}$ as relative coordinates to release the burden of the diffusion model to also predict the distribution center for different systems. For the lattice vectors, we introduce a reference lattice vector set $\bar{\mathbf{L}}^\star$ taken as the mean lattice vector set on the dataset, and diffuse the relative vectors $\mathbf{L} := \bar{\mathbf{L}} - \bar{\mathbf{L}}^\star$. For the carbon-atom coordinates, we use their relative coordinates w.r.t the unit cell center $\bar{\mathbf{z}} := \frac{1}{2}(\bar{\mathbf{l}}_x + \bar{\mathbf{l}}_y + \bar{\mathbf{l}}_z)$, which means $\mathbf{X} := \bar{\mathbf{X}} - \bar{\mathbf{z}}$. The diffusion-process variable is then defined as $\mathbf{R} := (\mathbf{X}, \mathbf{L})$. The diffusion-variable part of the input to the diffusion model still requires the corresponding absolute coordinates $(\mathbf{X}, \mathbf{L})$ of $\mathbf{R}$. For the descriptor part of the input, in addition to the number of carbon atoms N_{atom}, we also include the reference lattice vectors $\bar{\mathbf{L}}^\star$, which is an informative feature similar to the discussion in Supplementary Sec. A.4.

The DiG model first learns the (unconditional) distribution of the structures of carbon polymorphs from a dataset created by random structure search (RSS). We take the noise-prediction form to learn the model (c.f. Eq. (A5)). The training process is detailed in Alg. 8. The DiG model is then asked to generate structure samples conditioned on a given property value, specifically a desired band gap value c in our case. According to Eq. (8), this requires a property predictor/classifier. For this, we use a GNN model M3GNet [11], which provides the prediction for the band gap of a given structure in absolute coordinates. To evaluate a probability, we convert the regression task

into a classification task by discretizing an inclusive range of the band gap value into K intervals of length 1.0, represented by $\mathcal{I}_0 = [a_0, b_0], \cdots, \mathcal{I}_{K-1} = [a_{K-1}, b_{K-1}]$. The property c is then taken as the bin index. Using the predicted band gap value from M3GNet, we define the probability of a given c as:

$$q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}, \bar{\mathbf{L}}) := \frac{\exp\left(-\left|\text{M3GNet}(\bar{\mathbf{X}}, \bar{\mathbf{L}}) - \frac{a_c + b_c}{2}\right|\right)}{\sum_{k=0}^{K-1} \exp\left(-\left|\text{M3GNet}(\bar{\mathbf{X}}, \bar{\mathbf{L}}) - \frac{a_k + b_k}{2}\right|\right)}. \tag{A30}$$

This equation is used to construct the required conditional score $\mathbf{s}^{\theta}_{\mathcal{D},t}(\bar{\mathbf{X}}_t, \bar{\mathbf{L}}_t \mid c)$ following Eq. (8). The value $\frac{a_c + b_c}{2}$ is the target band gap for the interval $\mathcal{I}_c$.

Algorithm 8 Carbon structure score model training (single step)

Require: Noise-predicting model $\epsilon^{\theta}_{\mathcal{D},t}(\bar{\mathbf{X}}, \bar{\mathbf{L}})$ to be trained, descriptor $\mathcal{D} = (N_{\text{atom}}, \bar{\mathbf{L}}^{\star})$, where N_{atom} is the number of carbon atoms, and $\bar{\mathbf{L}}^{\star} = \{\bar{\mathbf{l}}^{\star}_x, \bar{\mathbf{l}}^{\star}_y, \bar{\mathbf{l}}^{\star}_z\}$ is the reference lattice vectors; randomly sampled time step $t \in [0, \tau]$, randomly sampled structure $(\bar{\mathbf{X}}^{(n)}_{\mathcal{D},0}, \bar{\mathbf{L}}^{(n)}_{\mathcal{D},0})$ from the dataset, where $\bar{\mathbf{X}}^{(n)}_{\mathcal{D},0}$ comprises coordinates of carbon atoms, and $\bar{\mathbf{L}}^{(n)}_{\mathcal{D},0} = \{\bar{\mathbf{l}}^{(n)}_{\mathcal{D},0,x}, \bar{\mathbf{l}}^{(n)}_{\mathcal{D},0,y}, \bar{\mathbf{l}}^{(n)}_{\mathcal{D},0,z}\}$ is the collection of lattice vectors.

1: Calculate the unit cell center $\bar{\mathbf{z}}^{(n)}_{\mathcal{D},0} := \frac{1}{2}(\bar{\mathbf{l}}^{(n)}_{\mathcal{D},0,x} + \bar{\mathbf{l}}^{(n)}_{\mathcal{D},0,y} + \bar{\mathbf{l}}^{(n)}_{\mathcal{D},0,z})$;
2: Let $\mathbf{X}^{(n)}_{\mathcal{D},0} := \bar{\mathbf{X}}^{(n)}_{\mathcal{D},0} - \bar{\mathbf{z}}^{(n)}_{\mathcal{D},0}$, $\mathbf{L}^{(n)}_{\mathcal{D},0} := \bar{\mathbf{L}}^{(n)}_{\mathcal{D},0} - \bar{\mathbf{L}}^{\star}$;
3: Sample the noise variable $\epsilon^{(\mathbf{X})}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{3N_{\text{atom}}})$ and $\epsilon^{(\mathbf{L})}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_9)$;
4: Let $\mathbf{X}^{(n)}_{\mathcal{D},t} := \alpha_t \mathbf{X}^{(n)}_{\mathcal{D},0} + \sqrt{1-\alpha_t^2}\epsilon^{(\mathbf{X})}_t$, and $\mathbf{L}^{(n)}_{\mathcal{D},t} := \alpha_t \mathbf{L}^{(n)}_{\mathcal{D},0} + \sqrt{1-\alpha_t^2}\epsilon^{(\mathbf{L})}_t$ (c.f. Eq. (A4));
5: Let $\bar{\mathbf{L}}^{(n)}_{\mathcal{D},t} := \mathbf{L}^{(n)}_{\mathcal{D},t} + \bar{\mathbf{L}}^{\star}$ which is structured as $\{\bar{\mathbf{l}}^{(n)}_{\mathcal{D},t,x}, \bar{\mathbf{l}}^{(n)}_{\mathcal{D},t,y}, \bar{\mathbf{l}}^{(n)}_{\mathcal{D},t,z}\}$;
6: Let $\bar{\mathbf{z}}^{(n)}_{\mathcal{D},t} := \frac{1}{2}(\bar{\mathbf{l}}^{(n)}_{\mathcal{D},t,x} + \bar{\mathbf{l}}^{(n)}_{\mathcal{D},t,y} + \bar{\mathbf{l}}^{(n)}_{\mathcal{D},t,z})$ and $\bar{\mathbf{X}}^{(n)}_{\mathcal{D},t} := \mathbf{X}^{(n)}_{\mathcal{D},t} + \bar{\mathbf{z}}^{(n)}_{\mathcal{D},t}$;
7: Evaluate the loss $\left\| \epsilon^{\theta}_{\mathcal{D},t}(\bar{\mathbf{X}}^{(n)}_{\mathcal{D},t}, \bar{\mathbf{L}}^{(n)}_{\mathcal{D},t}) - (\epsilon^{(\mathbf{X})}_t, \epsilon^{(\mathbf{L})}_t) \right\|^2$ (c.f. Eq. (A6));
8: Update the model parameter θ by performing an optimization step on the loss with respect to θ.

The sampling process starts from a standard-Gaussian sample as $\mathbf{R}$, and in each step $\mathbf{R}$ is converted to absolute coordinates using $\bar{\mathbf{L}}^{\star}$ and the corresponding unit cell center for the input to the model, and finally outputs the structure sample in absolute coordinates. Similar to the case of protein structure sampling as explained at the end of Supplementary Sec. A.2.1, for simulating the sampling process, instead of simulating the SDE in the fashion of Eq. (3), it achieves better results to simulate the equivalent ODE in Eq. (7). This is explained in Supplementary Sec. A.1.3. Under discretization, Eq. (7)

Algorithm 9 Sampling for carbon structure inverse design

Require: A trained noise-predicting model $\epsilon^{\theta}_{\mathcal{D},t}(\bar{\mathbf{X}}, \bar{\mathbf{L}})$ which decomposes as $(\epsilon^{(\mathbf{X}),\theta}_{\mathcal{D},t}, \epsilon^{(\mathbf{L}),\theta}_{\mathcal{D},t})$ according to its output channels, descriptor $\mathcal{D} = (N_{\text{atom}}, \bar{\mathbf{L}}^{\star})$ of the target system, where N_{atom} is the number of carbon atoms, and $\bar{\mathbf{L}}^{\star} = \{\bar{\mathbf{l}}^{\star}_x, \bar{\mathbf{l}}^{\star}_y, \bar{\mathbf{l}}^{\star}_z\}$ is the reference lattice vectors; a trained property classifier $q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}, \bar{\mathbf{L}})$, the desired property value c, guidance strength λ_{guide}.

1: Sample a noisy initial structure $\mathbf{X}_{\tau} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{3N_{\text{atom}}})$, $\mathbf{L}_{\tau} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_9)$;
2: **for** i in $N, \cdots, 1$ **do**
3: $t_{i-1} := \frac{i-1}{N}\tau$;
4: Let $\bar{\mathbf{L}}_{t_i} := \mathbf{L}_{t_i} + \bar{\mathbf{L}}^{\star}$ which is structured as $\{\bar{\mathbf{l}}_{t_i,x}, \bar{\mathbf{l}}_{t_i,y}, \bar{\mathbf{l}}_{t_i,z}\}$;
5: Calculate the unit cell center $\bar{\mathbf{z}}_{t_i} := \frac{1}{2}(\bar{\mathbf{l}}_{t_i,x} + \bar{\mathbf{l}}_{t_i,y} + \bar{\mathbf{l}}_{t_i,z})$;
6: Let $\bar{\mathbf{X}}_{t_i} := \mathbf{X}_{t_i} + \bar{\mathbf{z}}_{t_i}$ which is structured as $\{\bar{\mathbf{X}}_{t_i,a}\}_{a=1}^{N_{\text{atom}}}$;
7: Let $\beta_i := \frac{\tau}{N}\beta_{t_i}$, $\alpha_i := \prod_{j=1}^{i}\sqrt{1-\beta_j}$;
8: $\mathbf{X}_{t_{i-1}} := (2 - \sqrt{1-\beta_i})\mathbf{X}_{t_i} - \frac{\beta_i}{2\sqrt{1-\alpha_i^2}}\epsilon^{(\mathbf{X}),\theta}_{\mathcal{D},t_i}(\bar{\mathbf{X}}_{t_i}, \bar{\mathbf{L}}_{t_i}) + \lambda_{\text{guide}}\frac{\beta_i}{2}\nabla_{\bar{\mathbf{X}}_{t_i}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}_{t_i}, \bar{\mathbf{L}}_{t_i})$ (c.f. Eqs. (A31, A5, 8));
9: $\mathbf{L}_{t_{i-1}} = (2 - \sqrt{1-\beta_i})\mathbf{L}_{t_i} - \frac{\beta_i}{2\sqrt{1-\alpha_i^2}}\epsilon^{(\mathbf{L}),\theta}_{\mathcal{D},t_i}(\bar{\mathbf{X}}_{t_i}, \bar{\mathbf{L}}_{t_i}) + \lambda_{\text{guide}}\frac{\beta_i}{2}\big(\nabla_{\bar{\mathbf{L}}_{t_i}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}_{t_i}, \bar{\mathbf{L}}_{t_i}) + \frac{1}{2}\sum_{a=1}^{N_{\text{atom}}} \nabla_{\bar{\mathbf{X}}_{t_i,a}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}_{t_i}, \bar{\mathbf{L}}_{t_i})\big)$ (c.f. Eqs. (A31, A5, 8));
10: **end for**
11: Let $\bar{\mathbf{L}}_0 := \mathbf{L}_0 + \bar{\mathbf{L}}^{\star}$ which is structured as $\{\bar{\mathbf{l}}_{0,x}, \bar{\mathbf{l}}_{0,y}, \bar{\mathbf{l}}_{0,z}\}$;
12: Calculate the unit cell center $\bar{\mathbf{z}}_0 := \frac{1}{2}(\bar{\mathbf{l}}_{0,x} + \bar{\mathbf{l}}_{0,y} + \bar{\mathbf{l}}_{0,z})$;
13: Let $\bar{\mathbf{X}}_0 := \mathbf{X}_0 + \bar{\mathbf{z}}_0$;
14: **Return** $(\bar{\mathbf{X}}_0, \bar{\mathbf{L}}_0)$.

is written as:

$$\mathbf{R}_{t_{i-1}} = \mathbf{R}_{t_i} + \frac{\beta_i}{2}\Big(\mathbf{R}_{t_i} + \mathbf{s}^{\theta}_{\mathcal{D},t_i}(\bar{\mathbf{R}}_{t_i} \mid c)\Big),$$

where h is the discretization step size, $\bar{\mathbf{R}}_t := (\bar{\mathbf{X}}_t, \bar{\mathbf{L}}_t)$ is the absolute coordinates corresponding to $\mathbf{R}_t$, and the plus sign is due to the simulation is reversed in time. Also recall $\beta_i := h\beta_{t_i}$. Since h hence β_i is an infinitesimal, the weight for the $\mathbf{R}_{t_i}$ term can be formulated as: $1 + \frac{\beta_i}{2} = 2 - (1 - \frac{\beta_i}{2}) = 2 - \sqrt{1-\beta_i} + o(h)$, which gives an alternative up to $o(h)$ which is acceptable since the disretization itself has $o(h)$ error. This then recovers the ODE simulation in [35]. By leveraging Eq. (8), the simulation step becomes:

$$\mathbf{R}_{t_{i-1}} = (2 - \sqrt{1-\beta_i})\mathbf{R}_{t_i} + \frac{\beta_i}{2}\Big(\mathbf{s}^{\theta}_{\mathcal{D},t_i}(\bar{\mathbf{R}}_{t_i}) + \nabla_{\mathbf{R}_{t_i}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{R}}_{t_i})\Big),$$

where $q_{\mathcal{D}}(c \mid \bar{\mathbf{R}}_{t_i})$ is given by Eq. (A30). Finally, by involing Eq. (A5), we can express the generation process using the noise-prediction model $\epsilon^{\theta}_{\mathcal{D},t}$:

$$\mathbf{R}_{t_{i-1}} = (2 - \sqrt{1-\beta_i})\mathbf{R}_{t_i} - \frac{\beta_i}{2\sqrt{1-\alpha_i^2}}\epsilon^{\theta}_{\mathcal{D},t_i}(\bar{\mathbf{R}}_{t_i}) + \frac{\beta_i}{2}\nabla_{\mathbf{R}_{t_i}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{R}}_{t_i}). \tag{A31}$$

Note that the input to the $q_{\mathcal{D}}(c \mid \bar{\mathbf{R}}_{t_i})$ model is absolute coordinates while the gradient is taken w.r.t the relative coordinates $\mathbf{R}_{t_i}$. To conduct this conversion, note that the conversion is $\bar{\mathbf{L}} = \mathbf{L} + \bar{\mathbf{L}}^{\star}$, and $\bar{\mathbf{X}} = \mathbf{X} + \bar{\mathbf{z}}$ where $\bar{\mathbf{z}} := \frac{1}{2}(\bar{\mathbf{l}}_x + \bar{\mathbf{l}}_y + \bar{\mathbf{l}}_z)$ with $\bar{\mathbf{L}} = \{\bar{\mathbf{l}}_x, \bar{\mathbf{l}}_y, \bar{\mathbf{l}}_z\}$. So $\nabla_{\mathbf{X}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}, \bar{\mathbf{L}}) = \nabla_{\bar{\mathbf{X}}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}, \bar{\mathbf{L}})$, and $\nabla_{\mathbf{L}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}, \bar{\mathbf{L}}) = \nabla_{\bar{\mathbf{L}}} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}, \bar{\mathbf{L}}) + \frac{1}{2}\sum_{a=1}^{N_{\text{atom}}} \nabla_{\bar{\mathbf{X}}_a} \log q_{\mathcal{D}}(c \mid \bar{\mathbf{X}}, \bar{\mathbf{L}})$. In practice, the strength of the property guidance can be tuned for better performance, as is a common practice in machine learning [81]. We hence introduce a λ_{guide} parameter. Alg. 9 details the sampling process.

It is important to note that, in the conditional generation, the classifier can be fully decoupled from the training of the diffusion model. As a result, our model can serve to generate any demanded properties when provided with a corresponding property prediction model. This flexibility allows the approach to be adapted for various applications and desired properties, given an appropriate predictive model.

A.6 Accelerating Inference

The inference procedure of DiG can be viewed as gradually removing the noise from Gaussian random variables to sample clear conformations in an approximated equilibrium distribution. Although it exhibits several orders of magnitude speedup for the cases in Sec. 3 compared to traditional simulation methods, the inference is still potentially expensive since it generally needs to go over all time steps. Fortunately, the inference time can be further saved with recently developed methods [82–86]. For example, Ref. [84] shows that the analytic solution of the diffusion ordinary differential equations (sampling of DiG can be alternatively viewed as solving the corresponding diffusion ordinary differential equations) can significantly accelerate the inference, where high-quality samples can be drawn in around 10 steps, resulting in a further 50 to 100 folds speedup. These recent advances demonstrate the potential for even more efficient inference procedures in the context of DiG and similar models. By combining these methods with the existing framework, it becomes increasingly feasible to generate desired structures with specific properties in a faster and more efficient manner.

A.7 Evaluation methods

Protein Conformation Sampling

As detailed in Sec. 3.1, this study aims to demonstrate the applicability of using the DiG method to sample protein distributions. To this end, simulations of

two proteins from the SARS-CoV-2 virus are used to approximate their actual distributions in equilibrium states. The molecular dynamics (MD) simulation trajectories of the receptor-binding domain (RBD) of the spike protein and the main protease are extracted from a public dataset[2]. For RBD, there are 2995 independent MD simulations with 1.8 ms of aggregate simulation time. For main protease, the aggregated simulation time is 2.6 ms from 5688 independent trajectories. Both sets of simulations are performed in the constant pressure and temperature (NPT) conditions at 310 K and 1 atm pressure.

To facilitate the comparison between distributions obtained from MD simulations and DiG generations, time-lagged independent component analysis (TICA) is utilized to project the simulated structures onto a low-dimensional manifold [87, 88]. This projection enables the visualization of the conformational distribution in the low-dimension space, such as the one spanned by the two TICA coordinates (Fig. 2a). The TICA projection analysis is carried out as the following: first, the backbone conformation is featurized with the *cosine* and *sine* values of backbone torsion angles; then standard TICA projection is executed using PyEmma [89], with lag times of 10 ns and 2 ns for RBD and main protease respectively. We first parametrized the TICA transformation matrix to MD simulation structures to obtain the probability distribution as references, then the same transformation is applied to structures generated by DiG. From MD simulation trajectories, about 1.8 million structures of each system are used for TICA analysis. Furthermore, to reduce the influence of disordered regions at protein termini, the terminal residues are excluded during this TICA analysis and projection to the low-dimensional conformational space. For the distribution comparison in the reduced 2D space, we focus on the populated regions, which correspond to metastable states. The regions with very low probability in the distribution map are not included for detailed comparison, in order to focus on the functional relevant conformations.

For a further comparative analysis between DiG and atomistic MD simulation, representative structures for both proteins are obtained from MD simulations by clustering analysis. Initially, cluster centroid coordinates of all dominant meta-stable clusters in the 2D TICA space (Fig. 2a) are estimated for each protein. Following this, 1000 MD simulated structures near cluster centroids are sampled for each structure cluster. The first 16 TICA components of these structures are used as features to cluster the simulation structures into 4 sub-clusters using the K-Means algorithm. Finally, the centroid structure of the largest sub-cluster was extracted by MDTraj [90] and taken as the representative structure of the respective meta-stable state.

In order to assess the degree of conformity of the distributions, we compute quantitative metrics on the 2D TICA space. Particularly, we devise a $G \times G$ grid to uniformly cover the populated regions of the 2D TICA space, wherein the grid is labelled as positive if there exists at least one simulation structure within the corresponding TICA range. This grid is referred to as the "groundtruth"

[2] https://covid.molssi.org/simulations/

grid. With structures sampled by DiG or other sampling methods, we can construct a similar "sampled" grid following the same procedure.

In the coverage analysis in Supplementary Sec. E, we employ four distinct types of metrics, namely:

$$\begin{aligned}
\text{Accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \\
\text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}, \\
\text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\
\text{F1} &= \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.
\end{aligned}$$

In the above equations, the grid labelled as the "groundtruth" is considered to be true, whereas the one labelled as "sampled" is considered to be predicted. The values TP, FP, TN, and FN represent the values of true positives, false positives, true negatives, and false negatives, respectively.

To quantitatively assess the similarity of structures to their reference conformations, we employ two metrics: the template modeling score (TM-score) [91] and the root mean square deviation (RMSD). TM-score is a normalized measure of structural similarity between two conformations, with a score of 1 indicating a perfect match; and RMSD calculates the average distance between the paired atoms of two optimally superimposed structures. In our evaluations, we restrict our RMSD calculations to the alpha carbon atoms in protein structure comparison, and for all non-hydrogen atoms for ligand structure comparison. These metrics provide a quantitative means of gauging the accuracy of our protein conformation samples relative to their experimental counterparts.

The quality of DiG-generated structures was assessed using the TM-scores, by comparing each generated structure against crystal structures (6M0J for RBD and 6LU7 for main protease). For RBD, the mean value of TM-score is 0.84 and all structures have TM-score larger than 0.8, indicating highly similar structures; while in the case of main protease, the TM-score is more spread, with about 94% structures with TM-score > 0.5. We adopt a criterion suggested by [92], using TM-score $= 0.5$ as a cutoff to remove the structures that are dissimilar to the experimental model. For the downstream analysis, such as structure distributions, only structures with TM-score > 0.5 are used to reduce noises from incorrect predictions.

Catalyst-Adsorbate Sampling

After training the model to find different adsorption configurations, we traverse the initial positions of the adsorbate by shifting the original initial structure of adsorbate in the dataset along the x and y vectors of the unit cell. Specifically, we equally divide the unit cell in x and y dimensions into a grid of 15×15

points. Without changing the conformation and height in the z dimension of the initial structure of the adsorbate, we shift its coordinates in the x and y dimensions to match these 15×15 points, thus creating 15×15 different initial structures of the system. For each initial position, we use DiG to sample 10 structures. The sampled structures are then verified by DFT relaxation with VASP [51]. In VASP, we allow both the catalyst surface and the adsorbate to move, which is consistent with our model and the dataset. The structures generated by our model are close to the relaxed structures, as described in Sec. 3. For some initial positions, we are able to find multiple adsorption sites within the 10 sampled structures. Fig. 4 shows such a case, where two of the adsorption configurations are sampled from the same initial structure. Note that the training dataset contains only very short MD trajectories. The capability of our model is also limited by the dataset. With longer MD trajectories that traverse more structures, our model should be able to find more adsorption sites from an arbitrary initial structure. More discussions can be found in Supplementary Sec. F.

The probability density map of DiG is generated with single-atom adsorbates. To plot the map of probability density, we equally divide the unit cell along x and y dimensions as mentioned previously, resulting in a 20×20 grid. We place the adsorbate above each grid point, creating 20×20 structures for each fixed height of the adsorbate atom. We traverse 10 different heights for the adsorbate atom, ranging from 0 Å to 1 Å from the surface of the catalyst. Atoms in the catalyst surface are kept as the initial positions when density evaluation. In total, we obtain 4000 structures. We calculate the log-likelihood of our model on these structures, given an initial structure where the adsorbate atom is 2 Å from the catalyst surface above the center of the catalyst surface in the unit cell. The equation below summarizes the calculation of the probability density map:

$$p(x_i, y_j \mid x_0, y_0, z_0) = \max_k p_{\text{model}}(x_i, y_j, z_k \mid x_0, y_0, z_0),$$

where $i, j, k \in \{1, \cdots, 20\} \times \{1, \cdots, 20\} \times \{1, \cdots, 10\}$, $z_0 = 2$ Å is distance from the initial position of the atom to the highest point of the catalyst surface, and (x_0, y_0) is the center of the catalyst surface in the unit cell. Finally, in the probability density map, we plot the log values of the probabilities above. In the energy map from VASP, for each i, j we plot the negative of the energy of the relaxed structure, with the x_i, y_j, and the catalyst surface fixed. Starting from an initial height $z_k = 2$ Å from the catalyst surface, we used DFT to relax along the z dimension to obtain the minimum energy upon the grid point (x_i, y_i).

Property-Guided Structure Generation

To measure the ability of modeling the carbon polymorph structures, we use the `StructureMatcher` in the Pymatgen [54] package to calculate the ratio that a sampled structure matches a structure in the training dataset.

We use hyperparameters `stol=0.5`, `angle_tol=10` and `ltol=0.3` for the `StructureMatcher`, where `stol` is the tolerance for the displacement of atom positions, `angle_tol` controls the difference in lattice vector angles between the matched structures, and `ltol` is the tolerance for the difference in matched lengths of lattice vectors. The `get_rms_dist` method is used to calculate the RMSD, which is normalized by the average free length per atom.

Appendix B Model Details

B.1 Descriptors of Molecular Systems

We consider four types of molecular systems in previous sections: proteins, protein-ligand, catalyst-adsorbate, and carbon polymorphs. A descriptor $\mathcal{D}$ is used for each system that captures the relevant features of the molecular structure and can be processed by DiG. The descriptor $\mathcal{D}$ in the four systems is first processed into node representations $\mathcal{V}$ describing the feature of each system-specific individual element, and a pair representation $\mathcal{P}$ describing inter-node features. Note that in some systems, the descriptor $\mathcal{D}$ also contains structural features, which are treated as part of the node representation $\mathcal{V}$. The $\{\mathcal{V}, \mathcal{P}\}$ representation is the direct input from the descriptor part to the Graphormer model, as illustrated in Fig. 1.

For protein systems, we follow AlphaFold [1] and adopt a coarse-grained representation that uses the position of the alpha-carbon atom and the orientation of each residue (see Supplementary Sec. A.2.1). The node representation $\mathcal{V}$ is a sequence of feature vectors that are generated by the Evoformer module in [1], which takes as input the amino acid sequence and the multiple-sequence alignment (MSA) of the protein. The pair representation $\mathcal{P}$ is composed of two matrices: the first represents all pairwise interactions between residues, which is also produced by Evoformer, and the second represents the lengths of all residue pairs on the amino acid sequence. The node and pair representations are learnable embeddings in [1], but we hold them fixed in this work to avoid additional computational cost from fine-tuning the Evoformer. See more discussions about the limitation of fixing parameters of Evoformer can be found in Supplementary Sec. F.

For protein-ligand binding systems, we use an all-atom representation that includes atoms from both the protein and the ligand. The features are obtained following the method in [77]. To be specific, the node representation $\mathcal{V}$ for the protein part consists of the types of atoms around the binding pocket and also the positions of these atoms $\widetilde{\mathbf{R}}^{\star}_{\mathrm{Rec}}$ in the crystal structure of the protein, and the ligand part consists of a graph of the skeletal formula of the ligand. The pair representation $\mathcal{P}$ is also composed of two parts: one for the intra-molecular bonds and one for the inter-molecular interactions. The intra-molecular bonds are represented by feature embeddings of the chemical bonds between the atoms of the protein or the ligand, and the inter-molecular interactions are represented by feature embeddings of interactions like the hydrogen bonding between the protein and the ligand, as defined in [77].

For catalyst systems, we use an all-atomic representation that includes both the catalyst surface and the adsorbates. The node representation $\mathcal{V}$ consists of the types of atoms of the catalyst and adsorbed molecules as well as their positions $\bar{\mathbf{R}}^{\star}_{\mathrm{Cat}}, \bar{\mathbf{R}}^{\star}_{\mathrm{Ad}}$ in the initial structure from the OC20 dataset [10]. The features of the nodes are only embeddings of the atomic type and position, following the method in [80]. The pair representation $\mathcal{P}$ is only defined for the adsorbates, which consists of feature embeddings of the chemical bonds between the atoms of the adsorbates.

For carbon polymorphs, we use an all-atomic representation that only includes carbon atoms. The node representation $\mathcal{V}$ consists of the initial embedding of the carbon element. The only difference among the systems is the number of carbon atoms. There is no pair representation $\mathcal{P}$ for the carbon polymorphs, as the chemical bonds are not pre-defined and may change during the diffusion process.

B.2 Backbone Architecture

The deep learning models used in DiG are extended by our previously proposed Graphormer [2, 80], which is a Transformer-based graph neural network [93], and could efficiently capture the topological information while keep the powerful expressiveness from the Transformer architecture. The model is composed of a few concatenated so-called attention layers and feed-forward layers. Each attention layer takes the hidden node representation $\mathcal{H}$ as the input tokens of the Transformer and uses the pair representation $\mathcal{P}$ as a learnable attention bias for the attention mechanism. Formally, given a hidden node representation $\mathcal{H} = \{\mathbf{h}_1, \cdots, \mathbf{h}_I\}$, where I is the number of nodes, and a pair representation $\mathcal{P} = \{\mathbf{P}_{ij}\}_{i,j=1}^{I}$, where $\mathbf{P}_{ij}$ is the learnable embedding of the edge features from node i to node j, Graphormer computes the attention score $\mathbf{A}_{ij}$ from node i to node j as:

$$\mathbf{A}_{ij} = \frac{(\mathbf{h}_i \mathbf{W}^{(Q)})(\mathbf{h}_j \mathbf{W}^{(K)})^{\top}}{\sqrt{d}} + \mathbf{P}_{ij}^{\top} \mathbf{w}^{(\mathcal{P})}, \tag{B32}$$

where $\mathbf{W}^{(Q)}, \mathbf{W}^{(K)}$ are the "query" and "key" linear projections for the node representation, $\mathbf{w}^{(\mathcal{P})}$ is a learnable weight vector for the pair representation, and d is the dimension of the query and key vectors. The hidden node representation $\mathcal{H}$ in the first layer is $\mathcal{V}$. The attention bias term $\mathbf{P}_{ij}^{\top} \mathbf{w}^{(\mathcal{P})}$ enables the model to learn the importance of the pair representation for the attention mechanism. The attention score is then normalized by a softmax function over all nodes and used to compute the attention output following the invariant point attention mechanism in AlphaFold [1] for protein systems, or standard Transformer architecture [93] for other molecular systems.

Algorithm 10 Equivariant Vector Prediction

Require: Node 3D positions $\{\mathbf{R}_1, \cdots, \mathbf{R}_I\}$, node representation $\mathbf{H} = [\mathbf{h}_1^\top, \cdots, \mathbf{h}_I^\top]^\top \in \mathbb{R}^{I\times C}$, attention bias $\mathbf{E}^{\text{attn}} \in \mathbb{R}^{I\times I}$; linear projector matrices $\mathbf{W}^{(Q)}, \mathbf{W}^{(K)} \in \mathbb{R}^{C\times d}$, $\mathbf{W}^{(V)} \in \mathbb{R}^{C\times C}$, $\mathbf{w}^{(F)} \in \mathbb{R}^C$;

Calculate the relative positions $\mathbf{E}^{\text{rel}} \in \mathbb{R}^{I\times I\times 3} : \mathbf{E}^{\text{rel}}_{\imath\jmath} := \mathbf{R}_\imath - \mathbf{R}_\jmath$;

$\mathbf{Q} = \mathbf{H}\mathbf{W}^{(Q)}, \mathbf{K} = \mathbf{H}\mathbf{W}^{(K)}, \mathbf{V} = \mathbf{H}\mathbf{W}^{(V)}$;

$\mathbf{A} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} + \mathbf{E}^{\text{attn}}$;

Calculate $\mathbf{F} \in \mathbb{R}^{I\times 3} : \mathbf{F}_\imath := \sum_{\jmath=1}^{I} \text{softmax}(\mathbf{A}_{\imath:})_\jmath (\mathbf{w}^{(F)\top} \mathbf{V}_\jmath) \mathbf{E}^{\text{rel}}_{\imath\jmath}$;

Return F;

B.3 Structural Attention Biases

Besides the descriptor $\mathcal{D}$ input of the molecular systems, the Graphormer model also needs to process the geometric structure input $\mathbf{R}$ and produce a physically finer structure. To more informatively encode the geometric information during the diffusion process, we also introduce a structural representation for the input $\mathbf{R}$, which is used to help Graphormer to capture the spatial and rotational relationships among the nodes and refine the noisy structures to more physically realistic ones.

For all molecular systems in full-atom representation, we follow [80] to encode the Euclidean distance $d_{\imath\jmath}$ between the positions of node $\imath$ and node $\jmath$ as a bias term $b_\phi(d_{\imath\jmath})$, where ϕ is a learnable parameter. The distance encoding bias is added to the attention score in Eq. (B32) to modulate the attention based on the distance between nodes. For protein systems that use the coarse-grained representation $\mathbf{R} = (\mathbf{C}, \mathbf{Q})$ (see Supplementary Sec. A.2.1), we adopt the invariant point attention mechanism in [1] to construct the corresponding attention score, which has been shown to be indispensable for capturing the local rotational invariance feature of the protein structures.

B.4 Equivariant Graphormer

Due to the score model interpretation (gradient of log-density function), the output of the Graphormer model is required to be equivariant w.r.t the $\mathbf{R}$ input, which requires a proper design for processing the $\mathbf{R}$ input. To ensure the rotational equivariance of the Graphormer model, we add one equivariant attention layer [80] as the 3D vector output head, which produces geometric vectors that are equivariant to any rotation transformations in 3D Euclidean space on the input, as detailed in Alg. 10. Specifically, we first compute the attention matrix $\mathbf{A}$ and the transformed invariant features $\mathbf{H} = [\mathbf{h}_1^\top, \cdots, \mathbf{h}_I^\top]^\top$ as in the previous layers, and then the vector output is obtained by attentively aggregating the relative position information and the invariant features. Since we only apply scalar multiplication and linear combination operations on the vector features, the resulting vector $\mathbf{F}$ is naturally equivariant to the SO(3) group of rotations. Moreover, $\mathbf{F}$ is translation-invariant because it only depends on the relative positions between two nodes. This design strategy for

processing vector features is similar to those used in previous works on protein modeling [94] and quantum chemistry [95].

B.5 Periodic Boundary Condition

In catalyst-adsorbate systems and carbon polymorphs, atoms in a 3D unit cell are periodically repeated. Therefore, radius graphs with periodic boundary conditions are constructed to represent the systems, where each atom in one single cell (the centric cell) will connect with its neighboring atoms within a pre-defined cutoff distance. Since atoms are periodically repeated, the same atom in different cells may appear repeatedly in one graph as different nodes. To avoid that the same atom has different node representations in the network, typically a multi-graph will be constructed for message-passing neural networks (MPNNs), where one node represents one atom, and multiple edges between nodes represent the interactions with the same atom in different cells. In this way, information on neighboring atoms will be aggregated by MPNNs with multiple times through each edge.

Differently, message aggregation is done by attentively weighted sum on full graph in Graphormer, and interactions between atoms are encoded into spatial distance embeddings acting as attention bias. Multi-graph will lead to a summation of multiple biases in the distance embedding space, which might be projected to a new distance, and would not reflect multiple interactions with the same atom in different cells. Therefore, to reflect the multiple interactions correctly while enforce one representation for the same atom in different cells, we use a cross-attention sub-layer to aggregate information from all atoms in the radius graph into the atoms in the centric cell as shown in Alg. 11.

Algorithm 11 Handling Periodic Boundary Condition

Require: Atom positions in the centric unit cell $\widetilde{\mathcal{X}} := \{\widetilde{\mathbf{x}}_i\}_{i=1}^{I}$.
Require: Lattice vectors $\mathbf{l}_x, \mathbf{l}_y, \mathbf{l}_z$. Cutoff distance D_{cut}.
Require: Atom representation $\left\{\mathbf{h}(\widetilde{\mathbf{x}}_i) \mid \widetilde{\mathbf{x}}_i \in \widetilde{\mathcal{X}}\right\}$ in current layer.
Ensure: Atom representation $\left\{\mathbf{h}'(\widetilde{\mathbf{x}}_i) \mid \widetilde{\mathbf{x}}_i \in \widetilde{\mathcal{X}}\right\}$ after attention.

$\mathcal{X} \leftarrow \left\{\widetilde{\mathbf{x}} + l\mathbf{l}_x + m\mathbf{l}_y + n\mathbf{l}_z \mid l, m, n \in \mathbb{Z}, \widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}}\right\}$;
$\mathcal{X}_D \leftarrow \left\{\mathbf{x} \in \mathcal{X} \mid \exists \widetilde{\mathbf{x}}' \in \widetilde{\mathcal{X}},\ \|\mathbf{x} - \widetilde{\mathbf{x}}'\| \leqslant D_{\text{cut}}\right\}$;
$\forall \mathbf{x} \in \mathcal{X}_D, \widetilde{\mathbf{x}} := \mathbf{x} + l\mathbf{l}_x + m\mathbf{l}_y + n\mathbf{l}_z$, s.t. $\widetilde{\mathbf{x}} \in \widetilde{\mathcal{X}},\ l, m, n \in \mathbb{Z}$;
$\mathbf{A}_{ij} \leftarrow \frac{\left(\mathbf{h}(\widetilde{\mathbf{x}}_i)\mathbf{W}^{(Q)}\right)\left(\mathbf{h}_j(\widetilde{\mathbf{x}}_j)\mathbf{W}^{(K)}\right)^{\top}}{\sqrt{d}} + b_\phi\left(\|\mathbf{x}_i - \mathbf{x}_j\|\right),\ \forall \mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}_D$;
$\mathbf{h}'(\widetilde{\mathbf{x}}_i) \leftarrow \sum_j \text{softmax}(\mathbf{A}_{i:})_j \left(\mathbf{h}(\widetilde{\mathbf{x}}_j)\mathbf{W}^{(V)}\right)$.

The basic idea is that learnable node embeddings are only assigned to atoms in the centric cell, and the embeddings for the same atom appearing in neighboring cells are its replicas. The attention bias encoded from pair-wise

atom distances is used to tell replicas of atoms in different cells. Therefore, the representation of each node in the centric cell will be updated by the correlation and interaction with all atoms in the radius graph.

In modeling the carbon polymorphs, with the lattice vectors, we can expand the unit cell and handle periodic boundary conditions as described in Alg. 11. However, we find that explicitly encoding the vertices in the unit cell as tokens in the Graphormer encoder is very helpful for sampling physical structures. Thus, each vertex of the unit cell is treated as an atom with a special type in the model. In other words, for a structure with n atoms in the unit cell, the model will take $n + 8$ tokens, and expand the tokens according to the PBC handling method. Alg. 8 and 9 describe the details of the training and sampling processes, respectively.

Appendix C Training Details

C.1 Protein Conformation Sampling

Training Pipeline and Dataset

Our training process for the protein system consists of three stages: initialization, physics-informed diffusion pre-training (PIDP), and data-based training using simulation data. In the first stage which aims to provide a good initialization to stabilize PIDP training, we collect all experimental structures from the Protein Data Bank (PDB) [96] before December 25, 2020 as training data and employ Alg. 4 for model training. Such experimental structures are widely used in structure prediction methods, in which case a dataset is organized following the pattern $(\mathcal{D}, \mathbf{R}_{\mathcal{D}})$, where each amino-acid sequence $\mathcal{D}$ is paired with one experimental structure $\mathbf{R}_{\mathcal{D}}$. In contrast, to provide distributional information during the training of DiG, we organize the structures to construct a physical distribution dataset, in which each data point follows the pattern $(\mathcal{D}, \{\mathbf{R}_{\mathcal{D}}^{(n)}\}_{n=1}^{N_{\text{data}}})$ where each amino acid sequence $\mathcal{D}$ is paired with a set of structures $\{\mathbf{R}_{\mathcal{D}}^{(n)}\}_{n=1}^{N_{\text{data}}}$. To prepare this dataset, we adopt all the sequence identity clusters from PDB obtained by MMSeqs2 [97], and include all the available experimental structures for each cluster. Following AlphaFold [1], we filter out structures from PDB that have a resolution worse than 9 Å. This eliminates about 0.2% of structures. For proteins longer than 256 amino acids, we divide them into segments of length no longer than 256 amino acids. In each training step, we randomly draw clusters and structures within each drawn cluster with equal probability. We would like to remark that although this physical structure distribution is hard to verify to obey the equilibrium distribution, it can still provide rich information about the different modes of the equilibrium distribution.

In the second stage, to prepare the relevant structures for evaluating the PIDP loss, we run short MD simulations for about 1000 proteins, for which the details can be found in Supplementary Sec. D. We randomly pick 100 simulated structures for each protein as the relevant structures. The training

process follows Alg. 1. In the final stage, we use a simulation dataset consisting of the above simulation dataset, and 238 simulation trajectories randomly picked from the GPCRmd dataset [76]. The GPCRmd dataset contains short simulations of various classes of G protein-coupled receptors (GPCRs). The training process follows Alg. 4.

PIDP Training

As mentioned in the main text (Sec. 2), the structures $\{\mathbf{R}_0^{(m)}\}_{m=1}^M$ for evaluating the PIDP loss are ideally grid points (as in finite-element methods) spanning the structure space, but this is unaffordable since the number of grid points increases exponentially with the dimension of the space, which is typically exceedingly high for molecular systems. But we only need to supervise the model on a low-dimensional manifold of physically relevant structures (with low energy). Due to the grid point nature, these structures do not have to follow the equilibrium distribution, but only need to demonstrate the manifold. This then enables a wide range of affordable methods to prepare such structures, such as perturbation around experimentally observed structures [98], and short MD simulation structures. In practice, we found protein structures perturbed by [98] lead to overly large energy gradient which hinders effective optimization and cannot be easily mitigated by e.g. gradient clipping (see Supplementary Sec. F.2 for more details about the limitation of energy function). For this reason, we adopt structure samples from short MD simulation trajectories for PIDP training. This is much cheaper than generating structures following equilibrium distribution by long enough MD simulations. The short MD simulations provide structures that can be seen in a physical process thus demonstrating the relevant manifold, and the information of equilibrium distribution is provided by the energy function.

For evaluating the energy function (or its gradient, i.e., force field), we use OpenMM to compute the full-atom force using the Amber force field as $-\nabla E(\bar{\mathbf{R}})$, which also serves for calculating the coarse-grained forces $-\nabla_{\mathbf{C}} E$ and $-\nabla_{\mathbf{q}} E$ through Alg. 2. We use PDBFixer [99] to fix the input protein structure files before processed by OpenMM to avoid potential failures.

We find that the range of magnitude of the calculated forces varies drastically, which poses a significant challenge in optimizing the PIDP loss in Eq. (4). We hence retain only the structures that have a force magnitude within the smallest three orders of magnitude. To further address the optimization challenge, we also modify the loss terms for matching the score model at $t = 0$ to the force field in Eqs. (A14, A19) by only matching their directions.

We also find that for loss terms for $t > 0$ (or $i > 0$) in the PIDP losses Eqs. (A14, A19) , different sampling time steps t (or i) result in significant differences in the scale of the loss, ranging from 0 to 1×10^6. For stable and effective training, we rescale and clip these losses. Specifically, with $\ell_t^{(\mathbf{C})}$ and $\ell_t^{(\mathbf{Q})}$ denoting the PIDP loss terms at time step t of alpha-carbon coordinates and residue orientations, we find $\ell_t^{(\mathbf{C})}$ and $\ell_t^{(\mathbf{Q})}$ increase exponentially with t,

Table C2: Hyperparameters of protein model. Different hyperparameters are used in different stages.

Hyperparameter	Initialization	PIDP	Data Training
Model depth		12	
Hidden dim (Single)		768	
Hidden dim (Pair)		256	
Hidden dim (Feed Forward)		1024	
Number of Heads		32	
Optimizer		Adam	
Learning rate schedule		Inverse Square Root	
Peak Learning rate	1E-03	1E-05	1E-05
Dropout p	0.1	0.0	0.1
Adam (β_1, β_1)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Adam ϵ	1E-06	1E-06	1E-06
Weight decay	1E-02	1E-02	1E-02
Warmup ratio	0.06	0.06	0.06
Batch size	128	32	128
Diffusion steps N	500	500	500
$(\sigma_{\min}^{(\mathbf{C})}, \sigma_{\max}^{(\mathbf{C})})$	(0.1, 35)	(0.1, 35)	(0.1, 35)
$[\sigma_{\min}^{(\mathbf{Q})}, \sigma_{\max}^{(\mathbf{Q})}]$	(0.02, 1.65)	(0.02, 1.65)	(0.02, 1.65)
Boundary weight λ_1	-	5	-
Hutchinson N_{est}	-	20	-

with exponents $\rho^{(\mathbf{C})}$ and $\rho^{(\mathbf{Q})}$. To avoid an excessively large loss, we scale these losses by: $\tilde{\ell}_t^{(\mathbf{C})} := \ell_t^{(\mathbf{C})}/(\rho^{(\mathbf{C})})^{1-\text{clip}(t)}$, and $\tilde{\ell}_t^{(\mathbf{Q})} := \ell_t^{(\mathbf{Q})}/(\rho^{(\mathbf{Q})})^{1-\text{clip}(t)}$, where $\text{clip}(t) := \min(0.05\tau, t)$.

Furthermore, we find that a balanced combination of the losses in Eq. (A3) and Eq. (4) is essential for generating more physical structures after PIDP training.

These training methods, implemented to ensure more stable optimization, may compromise the accuracy of the distribution learned by the model. These limitations will be discussed further in Supplementary Sec. F.

Hyperparameter Choices

In protein training, we discretize the diffusion-process time variable $t \in [0, \tau]$ into $i \in \{0, 1, \cdots, N\}$ discrete time steps. The noise scales σ_i in Eq. (A11) and Eq. (A18) (or in Algs. 1 and 4 for training, and Alg. 5 for sampling) at the corresponding discretized time step t_i are taken in the form $\sigma_i = (\sigma_{\min})^{(1-t_i/\tau)}(\sigma_{\max})^{t_i/\tau}$. The parameters $\sigma_{\min}^{(\mathbf{C})}$ and $\sigma_{\max}^{(\mathbf{C})}$ for the diffusion process on alpha-carbon coordinates and $\sigma_{\min}^{(\mathbf{Q})}$ and $\sigma_{\max}^{(\mathbf{Q})}$ for the residue orientation are detailed in Supplementary Tab. C2, together with all other hyperparameters. To strike a balance between computational efficiency and performance, we train for at least one epoch at each stage and halt training when the rate of loss reduction noticeably decelerates.

Table C3: Hyperparameters of protein-ligand binding model used in CrossDocked datasset and MD dataset training.

Hyperparameter	CrossDocked & MD Training
Model depth	12
Hidden dim (Model)	768
Hidden dim (Feed Forward)	768
Number of Heads	32
Optimizer	Adam
Peak learning rate	0.0002
Warmup ratio	0.06
Learning rate schedule	Linear Decay
Dropout p	0.1
Adam (β_1, β_2)	(0.9, 0.98)
Adam ϵ	1E-08
Weight decay	0.0
Batch size	64
Diffusion steps N	500
Diffusion β schedule	Sigmoid
Diffusion β start	1E-07
Diffusion β end	0.02
EMA decay	0.9999
EMA fp32	true
Clip norm	10.0

C.2 Ligand Structure Sampling around Binding Sites

Two-stage training was performed in ligand sampling. The first stage employs CrossDocked [79] as a binding structure prediction task. The CrossDocked dataset contains docked conformations of varying quality, so we filter out all complexes whose RMSD between the docked and the experimental crystal structures is larger than 2.5 Å. The second stage employs the simulation data (Sec. D.1), where we set the threshold as 6 Å, and 1 ns as the sampling stride. For data-based training, we collect data from CrossDocked [79] and MD simulations to explore the most concerned part in the conformational space. The CrossDocked dataset contains docked conformations of varying quality, so we filter out all complexes whose RMSD between the docked and the experimental crystal structures is larger than 2.5 Å. For MD simulation data, we set the threshold as 6 Å, and 1 ns as the sampling stride. We conduct a quality screening on the simulation data, by filtering out trajectories that there are ligand's atoms around the protein laying within 5 Å. After filtering, the MD simulation dataset contains 1157 protein-ligand complex trajectories, and we split 80% for training, 10% for evaluation, and 10% for testing. All hyperparameters are listed in Supplementary Tab. C3.

C.3 Catalyst-Adsorbate Sampling

Our model captures the distribution of the structure of an adsorbate on a catalyst surface. DiG predicts the distribution conditioned on an initial structure, which is the first frame in a relaxation trajectory provided in the OC20 dataset [10]. To train DiG, we use the MD part of the OC20 dataset following Alg. 6. 20,000 systems of the MD dataset are separated for validation. Before this data-based training, the Graphormer model is first pretrained on the IS2RS task of OC20. Detailed hyperparameters for the IS2RS pretraining are listed in Supplementary Tab. C4. In the data-based training, we use a peak learning rate of 2×10^{-4}, maximum number of epochs 300, warm-up ratio 6%, a batch size of 64, and the number of diffusion steps $N = 5000$. The beta schedule follows a sigmoid form:

$$\beta_i = \frac{1}{1 + \exp\left(12(0.5 - i/N)\right)}(\beta_{\text{end}} - \beta_{\text{start}}) + \beta_{\text{start}},$$

with $\beta_{\text{start}} = 1 \times 10^{-7}$ and $\beta_{\text{end}} = 2 \times 10^{-3}$ where $i \in \{0, \cdots, N\}$ is the diffusion time step. The training is stopped after 86 epochs. We use a cutoff value of 6 Å for PBC handling. Supplementary Tab. C4 summarizes detailed hyperparameters for training of DiG for catalyst-adsorbate sampling.

Table C4: Hyperparameters of backbone model for pretraining on Open Catalyst IS2RS and training on Open Catalyst MD dataset.

Hyper Parameter	IS2RS pretraining	MD training
Model depth	12	
Hidden dim (Model)	768	
Hidden dim (Feed Forward)	768	
Number of Heads	32	
Optimizer	Adam	
Learning rate	0.0002	
Warmup ratio	0.06	
Learning rate schedule	Linear Decay	
Dropout p	0.1	
Adam (β_1, β_2)	(0.9, 0.98)	
Adam ϵ	1E-08	
Weight decay	0.0	
PBC Cutoff	6.0	
Batch size	1024	64
Diffusion steps N	-	5000
Diffusion β schedule	-	Sigmoid
Diffusion β start	-	1E-07
Diffusion β end	-	0.002

C.4 Property-Guided Structure Generation

For training the DiG on an unconditional distribution, we use 15,697 structures of carbon polymorphs generated from ab initio random structural search (RSS) at the DFT level (PBE/plane-wave basis, with an energy cutoff of 520 eV) with a range of number of atoms from 2 to 24, following the method in [52]. Only the relaxed structures, i.e., the final frames in relaxation processes, are taken for training. We remark that for the inverse design task, the distribution of the structures at local energy minima does not follow an equilibrium distribution. However, the primary goal here is to generate structure candidates with reasonable stability and targeted properties. In this context, the relaxed structures from random structure search can well represent the low energy structure manifold. The number of training epochs is 50,000, with a peak learning rate 2×10^{-4}, batch size 4096, and a warm-up ratio 6%. For the reference lattice vector set $\bar{\mathbf{L}}^{\star}$, we use the mean lattice vector set over the dataset, which is close to three orthogonal vectors with a length of 4 Å. As the initial structure in the catalyst-adsorbate model, this reference lattice vector structure is also encoded into the model with an additional attention bias term. Tab. C5 summarizes the hyperparameters for training the diffusion model for property-guided structure generation. We use a cutoff value of 20 Å for PBC handling.

Table C5: Hyperparameters of diffusion model training for property-guided structure sampling.

Hyperparameter	Data Training
Model depth	12
Hidden dim (Model)	768
Hidden dim (Feed Forward)	768
Number of Heads	32
Optimizer	Adam
Learning rate	0.0002
Warmup ratio	0.06
Learning rate schedule	Linear Decay
Dropout p	0.1
Adam (β_1, β_2)	(0.9, 0.98)
Adam ϵ	1E-08
Weight decay	0.0
PBC Cutoff	20.0
Batch size	1024
Diffusion steps N	500
Diffusion β schedule	Sigmoid
Diffusion β start	1E-07
Diffusion β end	0.02

Appendix D Molecular Simulation and Energy Evaluations

D.1 Molecular Dynamics Simulation for Protein-Ligand Complexes

We generate MD simulation data for complex systems selected from the PDBbind v2020 [100] using an automatic pipeline called protocolGromacs[3]. It utilizes GROMACS [101] as the backend engine with a common simulation setting for all complexes, providing a capability of high-throughput MD simulations. Specifically, this pipeline comprises four stages: preparation, minimization, equilibration, and production simulations. In the system preparation stage, a protein topology is generated with pdb2gmx with the amber99sb-ildn [102] force field with the tip3p explicit water model; the ligand parameter and topology are generated with acpype [103]. For cases with missing atoms/residues in the PDB files, PDBFixer [99] is applied to complete the molecules. Then, a cubic simulation box is used with a minimum distance of 1.2 nm between the protein-ligand complex and the box boundaries. Finally, a pre-equilibrated system of 216 water molecules is repeated over the simulation box to provide the solvated environment. To neutralize charged systems, appropriate ions (Na^+ or Cl^-, depending on the net charge of the solute molecules) are applied by replacing randomly selected water molecules. Once the simulation box is prepared, an energy minimization process is carried out to remove the atomic clashes and optimize the geometry of all molecules. In the equilibration simulation stage, a thermostat is applied to heat the system from 0 to 300K within 100 ps. The heated system is then further equilibrated to 1 bar in an NPT ensemble for another 100 ps. During the equilibration stage, the bonds for molecules are constrained. For the final production, the leap-frog algorithm [104] is used for integrating Newton's equations of motion and a Particle Mesh Ewald (PME) [105] method is used for calculating long-range electrostatic interactions. The LINCS [106] algorithm is adopted for resetting all bonds to their correct lengths after an unconstrained update. Finally, the production is performed for 100 ns with an integration time step of 2 fs.

Following the above protocol, we generate MD simulation trajectories for 1500 protein-ligand complexes. To facilitate model training, each 100-ns simulation trajectory from the production run is divided into segments of length 1 ns, resulting in 100 trajectory segments for each complex system. The protein-ligand complex is mapped to the center of the primary simulation box by applying periodic boundary conditions to ensure the integrity and connectivity of the molecules.

[3] https://github.com/tubiana/protocolGromacs

D.2 Energy Evaluations in Physics-Informed Diffusion Pre-training

For physics-informed diffusion pre-training (PIDP), the energy and gradients are evaluated using OpenMM [74] following the settings used in Folding@home [44]. The amber14sb force is used for the ablation study in Supplementary Sec. E.3. The Generalized Born solvent model [107] is used for solvation energy calculation.

D.3 Density Functional Theory Computation

We use DFT for the grid search of adsorbate configurations on catalyst surfaces, for verifying the adsorbate configuration distributions predicted by DiG as shown in Fig. 4. They are carried out using VASP6.3 with setups compatible with OC20 [10]. Specifically, periodic boundary conditions and projector-augmented wave pseudopotentials are adopted with plane-wave electron kinetic energy cut-off of 350 eV. Generalized gradient approximation and the revised Perdew-Burke-Ernzerhof (RPBE) functional are employed [108, 109]. The Monkhorst-Pack grid is used to sample the reciprocal space. For the electronic degree of freedom, the convergence criteria for self-consistent computations is set to 1×10^{-3} eV/atom. For ionic degree of freedom, the relaxations are carried out only on the atomic coordinates with the lattice being fixed. Convergence is considered to be reached when the Hellmann-Feynman forces are smaller than 0.02 eV/Å.

Appendix E Additional results

E.1 Protein Conformation Sampling

Table E6: Protein systems utilized in this paper. Reference structure 1 is denoted in cyan, while reference structure 2 is denoted in brown in Fig. 2b.

Protein	Ref. 1	Ref. 2	TMscore
Adenylate Kinase	4ake (chain A)	1ake (chain A)	0.6899
Lmrb	DEER-AF	6t1z (chain A)	0.7600
human B-Raf kinase	6uan (chain A)	3skc (chain A)	0.9235
D-ribose	3dri (chain A)	1urp (chain A)	0.7187

We present a list of proteins employed to demonstrate the efficacy of DiG in generating multiple conformations in Fig. 2b in Table E6. The table provides a detailed overview of the protein systems utilized in this study, including the reference structures denoted in cyan and brown, respectively, and the conformational differences between the two reference structures, measured in TMscore.

For both RBD and main protease proteins, there are some structures in the PDB dataset used for DiG model training. We project those structures onto the reduced 2D space spanned by the first two TICA coordinates, and show the results in Supplementary Fig. S1. Clearly, multiple states are observed in the maps indicated by the diamonds, but these structures together only represent a small fraction of the structure space. In contrast, the DiG-generated structures show much broader overlaps with MD simulations in the structure space. In the case of RBD, DiG even predicted a new region (lower right, cluster-IV in the main text), which has no experimental determined structure. This particular region was significantly sampled by MD simulation. We can observe clear correspondence for the regions sampled by DiG and MD simulations. The diverse structures reveal more information about the functional states of proteins.

Besides the qualitative comparison of the distributions generated by DiG and those sampled by MD simulations. We provide a detailed quantitative measurement of the overlaps of explored regions by these two methods. Taking the regions with MD simulation structures as references, the coverage analysis is carried out following a standard binary classification approach (see Supplementary Fig. S1). The distributions are first converted to masked binary maps (divided to 50×50 regions), then the two sets of binary maps (DiG results vs. MD simulation results) are compared. The accuracy, precision, recall, and F1-score are computed for each case, at different sampling data sizes. There are 50,000 structures generated by DiG for each protein. We see that the coverage (recall) increases as more structures are included for the analysis, while the precision level is kept at high levels. In order to compare with the MD simulation data, we took two approaches to draw samplings: (1) take structures consecutively from simulation trajectories; (2) take structures randomly from all simulation trajectories (i.i.d sampling). We observed that the sampling coverage and efficiency of DiG are better than MD simulations.

For two representative ligand-protein systems shown in Fig. 3b, structures from 100 ns MD simulations are superposed to show the variation in ligand structures (see Supplementary Fig. S2).

The ligand structure generations are carried out for a set of 16 proteins, each paired with various numbers of ligands. In total, there are 409 ligand-protein systems in this testing dataset. In Supplementary Fig. S3, five proteins, each with four different ligands, are shown to illustrate the best structures generated by DiG. The ligand binding poses and atomic structures generated by DiG exhibit diversity and are correlated with the characteristics of protein pockets.

E.2 Catalyst-Adsorbate Sampling

Supplementary Fig. S4 shows top and front views of all adsorption configurations generated by DiG, which covers all the adsorption configurations found by DFT relaxation for this system by traversing initial positions of the adsorbate. Note that for most systems, with a very short MD trajectory as provided

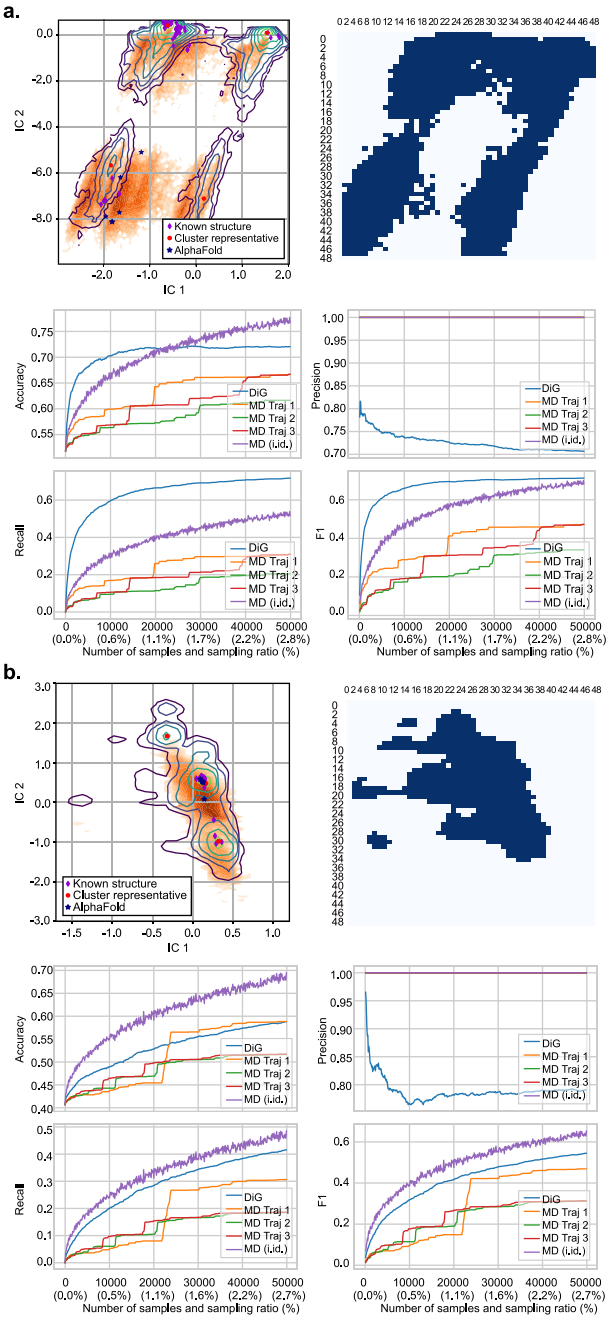

Fig. S1: Protein structure distributions and sampling coverages.

Fig. S1: **a.** Results for the RBD of SARS-CoV-2 spike protein. Experimentally determined structures are mapped to the reduced 2D space, indicated using the purple diamond symbols. On the top right panel, the same space is divided into 50x50 grids, which are classified into explored (blue) and unexplored (white) sub-regions, depending on the presence of MD simulation structures. The accuracy, precision, recall, and F1-score are shown as a function of sampling size (or the ratio to the whole dataset). The cluster in the lower-right region has no experimental structures, indicating a new state revealed DiG, which is consistent with MD simulations. **b.** Results for the main protease following the same analysis protocol and representations. In both plots **a** and **b**, blue star symbols indicate the AlphaFold predicted structures in the 2D space, and red circles show the cluster centers of MD simulation structures.

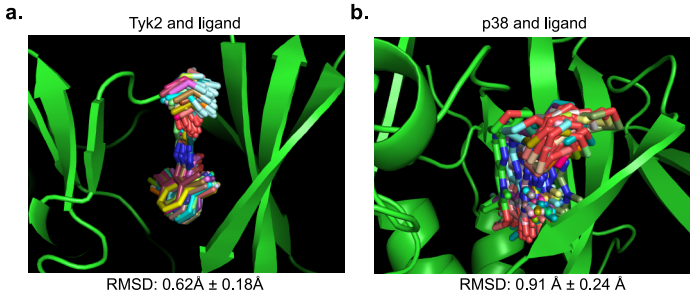

Fig. S2: **Ligand structures observed in MD simulations. a.** For the case of Tyk2 target, the binding pocket is deep and well confined, MD simulations show highly similar ligand structures and binding poses. The RMSD values compared to the crystal structure is small. **b.** the MD simulation results are shown for the case of P38 protein. Similar to the DiG results, the ligand structure exhibits larger variations compared to the cases of tyk2. For both proteins, the same ligands shown in Figure 2 in the main text are used in the simulations, and the simulation duration is 100 ns for both systems.

in the dataset, it can only cover one or two structures close to these relaxed configurations. Thus, the result shows the ability of DiG to generate to unseen systems from very short MD trajectories in the training set.

E.3 Ablation Study

We conduct an ablation study to investigate the effect of various components in the training pipeline of DiG for protein systems. The training pipeline consists of three stages: (i) initialization from experimental data, (ii) physics-informed diffusion pre-training (PIDP), and (iii) training with simulation data.

The main protease of SARS-CoV-2 is selected as a case study since it has long simulation trajectories (2.6 ms) [44] and has been studied in Section 3.1.

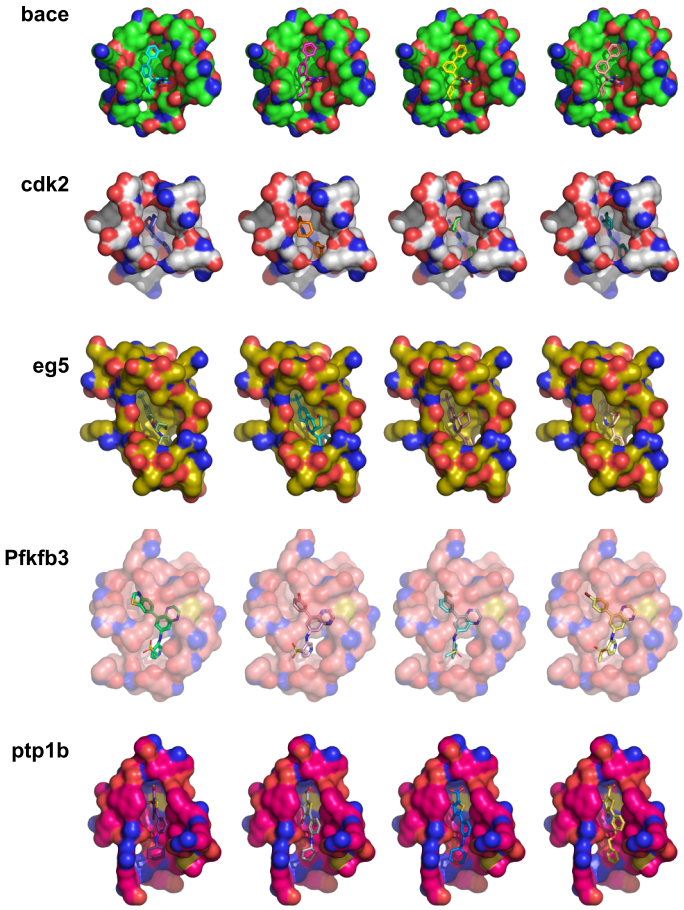

Fig. S3: **Ligand structure generation in the binding pocket of target proteins.** The target names are indicated in the figure, with each row showing different ligands with its best binding poses to the same target protein. Here, best binding poses is defined as the most similar structure to the experimental observations.

We investigate the importance of each stage of the training pipeline by evaluating the quality of structures sampled at different stages. The quality is measured by comparing the torsion angle distribution of sampled structures with that of the simulation trajectories. We use Ramachandran plots to visualize the distribution of torsion angles of the 6 amino acids corresponding to the first 10 TICA components of the simulation trajectories [110].

Supplementary Fig. S5 summarizes the results of the ablation study. We obtain approximately 100,000 filtered results from each of the three stages. The blue area represents the ground-truth simulation distribution, which encompasses a relatively large and diverse range of torsion angles, reflecting the

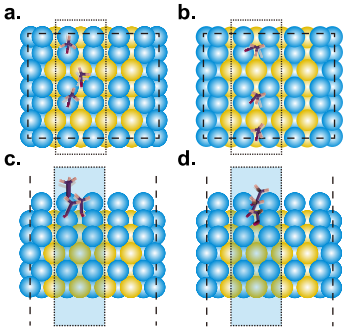

Fig. S4: Additional Results for catalyst surface adsorption. All adsorption configurations found by DiG, with the configurations from model in color and the configurations from grid search with DFT in white. The number of adsorption sites is 6 in total. We divide all the sites into 2 groups in (a)(c) and (b)(d), and show both the top view in (a)(b) and the front view in (c)(d).

dynamic and flexible nature of the protein. Supplementary Fig. S5a displays the distribution of structures sampled by DiG initialized from experimental data, i.e., the PDB protein structure dataset [96] (see Supplementary Sec. C.1). The distribution is highly concentrated on a single point, indicating that the sampled structures are only fit to experimental structures and do not capture the protein's dynamics. Supplementary Fig. S5b illustrates the distribution of structures sampled by DiG after PIDP, which improves the generated distribution towards the equilibrium distribution using the energy function. We find that PIDP training may generate some failure cases with very high RMSDs or low TMscores compared to the crystal structure. Therefore, we filter out results with RMSD higher than 10 Å and TMscore lower than 0.6.

However, PIDP alone is still insufficient to fully capture the equilibrium distribution. Supplementary Fig. S5c displays the distribution of structures sampled by DiG after further training with simulation data, where simulation structures serve as direct signals to supervise DiG's learning. The distribution becomes even more similar to the equilibrium distribution, demonstrating DiG's ability to learn from simulation data and generate realistic and diverse structures.

All sampled structures at various stages of the training pipeline are physically reasonable and are similar to low energy values, as verified by the structural quality metrics. The results show that DiG can effectively combine information from both energy function and simulation data to produce high-quality structures reflecting the protein systems' equilibrium distribution.

E.4 Reproducibility

In this section, we investigate the reproducibility of DiG training, under different initialization, and different random seeds which affect the order of data batches fed into the model during the training. We conduct this experiment

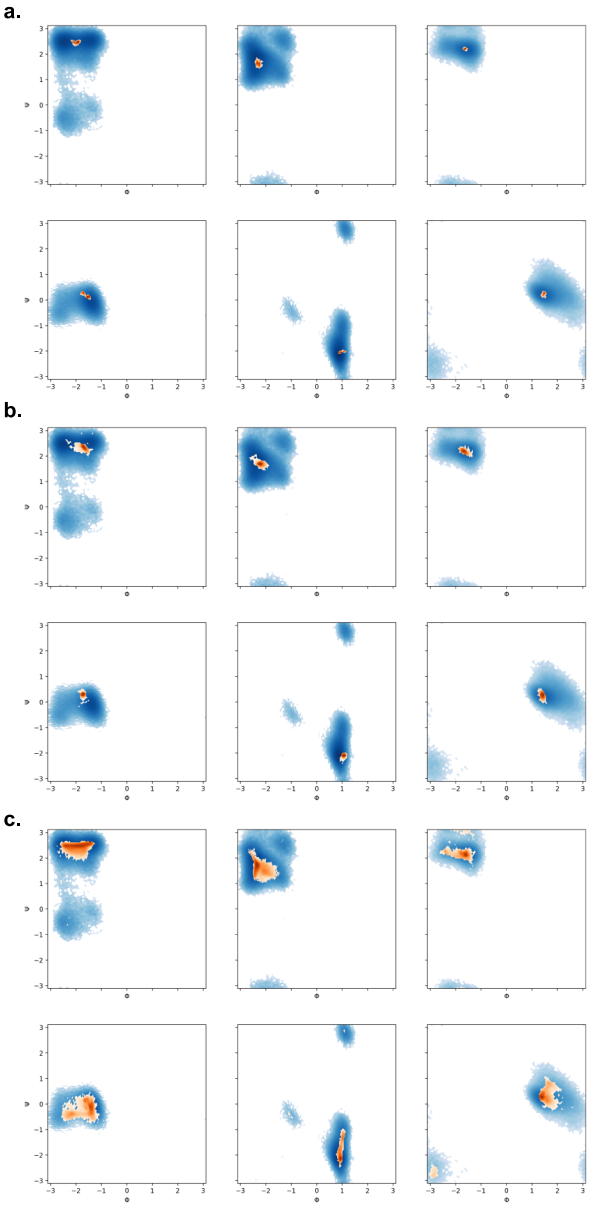

Fig. S5: Ramachandran plots of the sampled structures at different stages of the optimization process (initialization, PIDP, and simulation data training) compared with the reference MD structures.

on the protein-ligand systems and compare the distribution of the generated

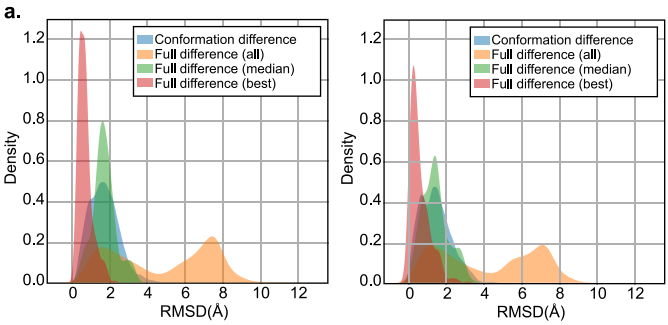

Fig. S6: Reproducibility experiments. Taking the ligand structure generation as an example, different DiG models were trained by varying training parameters. The results of the two trained DiG models show high similarity in terms of ligand structure differences compared to crystal structures.

ligand structures with respect to the crystal structures. Fig. S6 shows the histograms of RMSD statistics for ligand structures generated by DiG. The left panel is identical to Fig. 3a, where the results are obtained from the model checkpoint with the training pipeline described in Supplementary Sec. A.3, which is used to generate all results for protein-ligand systems in this paper. The initialization of this model is from pre-training on the CrossDock dataset. The right panel shows the results from another model checkpoint without any pre-training but a random initialization. Other hyperparameters except the random seeds are kept the same in the two training processes. We observe that the distributions from the two model checkpoints are very similar, suggesting that DiG training is robustly reproducible.

Appendix F Limitations

F.1 Limitations on Data Quantity and Quality

One of the major challenges and limitations of DiG is the scarcity of data for training and evaluating the deep learning models for equilibrium distribution prediction. The ground truth data of equilibrium distribution of different molecular systems are not easily available, as they require massive computational resources and time to generate by molecular dynamics simulation or other methods. Therefore, we only have access to very little data for some molecular systems, and the data quality and quantity may not be sufficient to support the learning and generalization of DiG.

For example, for the catalyst systems, we use the Open Catalyst dataset [10], which contains DFT-based molecular dynamics simulations of catalyst-adsorbate systems for only 80 or 320 femtoseconds. However, this simulation time is too short to capture the dynamics and transitions of the systems, and the structures may not move significantly from their initial positions. Thus, we need to traverse initial positions of the adsorbate to find out

all the adsorption configurations within a unit cell. With longer MD trajectories, the model should be able to sample more adsorption configurations from a single initial position of adsorbate. Moreover, the adsorption configurations in the MD trajectories, which start from the relaxed configurations, tend to have low energies, and high-energy configurations are rare in the dataset. Thus, the density for high-energy configurations estimated by the learned DiG may be inaccurate.

For property-guided structure generation, we use a dataset of 15,697 carbon crystal structures [53] to train the model, which is also very limited compared to the huge space of possible carbon polymorphs. The generative model trained on such data may not be able to recover all the structures in the dataset, let alone generalize to unseen carbon polymorphs. For example, our conditionally generated structures (including 2, 4, 6, 8 carbon atoms per unit cell) only match 88.33% of the structures in the dataset with the same numbers of atoms, using the `StructureMatcher` from Pymatgen.

For the protein conformation sampling, we collect MD simulations of about only 1000 proteins, each with 100 nanoseconds of simulation time. However, this amount of data may not be enough to cover the diversity and complexity of different protein structures and functions, especially for large and complex proteins that may have longer time scales and more energy barriers for conformational changes. Furthermore, the desired equilibrium distribution for protein model training is not well represented by the available data. Although some simulated data have sufficient length to approximate the equilibrium distribution, they are too scarce to support the neural network models with robust generalization and practical accuracy. Therefore, we resort to using a larger number of experimental structures, such as the PDB dataset, as the initial training data for the models. However, these data influence the final distribution learned by the models. We observe that these data lead to more accurate structures and better generalization, but also to a more concentrated learned distribution. Besides the issue of simulation length, the accuracy of simulation may also be problematic. We find that some structures simulated by molecular dynamics in some systems deviate too much from the experimental structures in terms of structural accuracy. Moreover, in some systems molecular dynamics is highly sensitive to the initial state, and different initial states can result in different distributions in practice. These factors compromise the use of simulated data as approximations of the equilibrium distribution and cause the model to learn inaccurate distributions.

Similar issues also exist in protein-ligand training, where the simulation time does not warrant equilibrium distributions. The simulation trajectories used for ligand-structure model training are limited to 100 ns. Yet, we observed ligand dissociation from the pocket in some of the systems, which were excluded from model training. Moreover, our training set only covers a small fraction of systems. In the CrossDocked and our self-generated MD simulation datasets, there are about only 1000 unique proteins, which may affect the generalization of our model.

The data scarcity also affects the evaluation of DiG, as we do not have enough ground truth data of equilibrium distribution to compare with the predictions of DiG. Therefore, we have to rely on indirect metrics, such as the energy function, the structural quality metrics, or properties, to measure the quality and diversity of the generated structures. However, these metrics may not fully capture the accuracy and reliability of the equilibrium distribution prediction, and may have some limitations or biases themselves. For example, the structural quality metrics may not account for the dynamic and stochastic nature of the molecular structures, and may have some dependencies on the reference structures or the alignment methods. The property prediction may not be sensitive to the subtle changes or variations of the molecular structures, and may have some noise or uncertainty in the measurements or the models.

Therefore, we acknowledge that data scarcity is a serious limitation for DiG, and we hope that more and better data of equilibrium distribution of molecular systems can be generated and shared in the future, to enable more robust and reliable learning and evaluation of DiG and other equilibrium distribution prediction methods.

Data limitations also affect the training of the property predicting model $q_{\mathcal{D}}(c \mid \mathbf{R})$, which guides the structure generation based on properties. In the sampling process for a desired property value c as described by Eqs. (3, 8) , the structures in early stages (i.e., large i or t) are nearly random noise, for which the predictor model $q_{\mathcal{D}}(c \mid \mathbf{R})$ are not typically trained on, making its contribution $\nabla_{\mathbf{R}} \log q_{\mathcal{D}}(c \mid \mathbf{R})$ less controlled. This may not be as harmful as it appears, since even with an oracle property predictor, what the early stage of sampling does is still refining the random structure to physically reasonable ones in which the predictor model contribution $\nabla_{\mathbf{R}} \log q_{\mathcal{D}}(c \mid \mathbf{R})$ does not dominate. The effect of this process also largely aligns with the desired property since for which a random structure is unlikely to achieve. However, we do observe that in some cases small perturbance to the structure causes significant changes in the band gap prediction using the M3GNet predictor, which makes sampling structures with demanded band gap more challenging. This also adds to the evidence of overfitting of the predictor to the limited stable structures. We also find that the model produces more unphysical structures that violate the geometric or energetic constraints when conditional generation. For example, a conditionally generated structure close to a graphite may not have perfect bond angles of exactly 120°. The band gap predictor model is trained on stable carbon polymorphs. But structures in the denoising process can be quite unstable. Thus the gradients from the predictor used to guide the denoising process may be inaccurate and does not always lead to physical structures. Adding guidance from an energy prediction model in the conditional generation process may guide to more physical structures. Training the property predictor with more abundant carbon polymorphs can also improve the quality of conditionally generated structures. Thus, it would still be helpful to generate labeled data on more noisy structures and train the property

predictor model on them, so that the model could take effect earlier in the sampling process to better guide the structure to the desired property.

F.2 Limitations on Energy Function

In applications involving proteins, we adopt the common choice of a coarse-grained representation for proteins to reduce the dimensionality of the problem while maintaining most of the structural features. This nevertheless incurs challenges from the energy function (equivalently, force field) side: we have to convert a full-atom force field to the coarse-grained level. This is required in the PIDP training as shown in Eqs. (A14, A19) , while employing an established coarse-grained force field is neither suitable since it is unnecessarily the coarse-grained version of the full-atom force field used in the simulation to generate the dataset. Such a conversion is conducted in Alg. 2, but this is not precise for coarse-graining for statistical use. Specifically, if denoting the invertible transformed full-atom coordinate $\bar{\mathbf{R}}$ and energy function as $(\mathbf{R}_{\mathrm{CG}}, \mathbf{R}_{\mathrm{FG}})$ and $E(\mathbf{R}_{\mathrm{CG}}, \mathbf{R}_{\mathrm{FG}})$ where $\mathbf{R}_{\mathrm{CG}}$ denotes the coarse-grained coordinates ($(\mathbf{C}, \mathbf{q})$ in Alg. 2) and $\mathbf{R}_{\mathrm{FG}}$ the fine-grained details ($\mathbf{X}$ excluding $\mathbf{C}$ in Alg. 2), the required coarse-grained energy (equilibrium free energy) under temperature T would be:

$$E_{\mathrm{CG}}(\mathbf{R}_{\mathrm{CG}}) = -k_{\mathrm{B}}T \log \int \exp\left\{-\frac{E(\mathbf{R}_{\mathrm{CG}}, \mathbf{R}_{\mathrm{FG}})}{k_{\mathrm{B}}T}\right\} \mathrm{d}\mathbf{R}_{\mathrm{FG}}. \tag{F33}$$

In practice, this integral is hard to evaluate, and is a long-standing problem in statistical mechanics and Bayesian statistics. Even in the case of Alg. 2 where we have access to the full-atom coordinates of a query structure, the estimation is still an approximation. To see this, the gradient (negative force) of Eq. (F33) can be written as:

$$\begin{aligned}
\nabla_{\mathbf{R}_{\mathrm{CG}}} E_{\mathrm{CG}}(\mathbf{R}_{\mathrm{CG}}) &= \mathbb{E}_{p_T(\mathbf{R}_{\mathrm{FG}} | \mathbf{R}_{\mathrm{CG}})}[\nabla_{\mathbf{R}_{\mathrm{CG}}} E(\mathbf{R}_{\mathrm{CG}}, \mathbf{R}_{\mathrm{FG}})], \\
p_T(\mathbf{R}_{\mathrm{FG}} \mid \mathbf{R}_{\mathrm{CG}}) &:= \frac{\exp\left\{-\frac{E(\mathbf{R}_{\mathrm{CG}}, \mathbf{R}_{\mathrm{FG}})}{k_{\mathrm{B}}T}\right\}}{\int \exp\left\{-\frac{E(\mathbf{R}_{\mathrm{CG}}, \mathbf{R}_{\mathrm{FG}})}{k_{\mathrm{B}}T}\right\} \mathrm{d}\mathbf{R}_{\mathrm{FG}}},
\end{aligned}$$

so in principle, the coarse-grained gradient is an average of the full-atom gradient over samples from $p_T(\mathbf{R}_{\mathrm{FG}} \mid \mathbf{R}_{\mathrm{CG}})$. Under this perspective, the rigid body assumption that Alg. 2 is based on can be understood as assuming $p_T(\mathbf{R}_{\mathrm{FG}} \mid \mathbf{R}_{\mathrm{CG}})$ only concentrates on one value of $\mathbf{R}_{\mathrm{FG}}$ (i.e., a Dirac delta distribution), meaning $\mathbf{R}_{\mathrm{FG}}$ can be uniquely determined from the given $\mathbf{R}_{\mathrm{CG}}$. In the algorithm, this $\mathbf{R}_{\mathrm{FG}}$ is provided from the corresponding full-atom coordinates. This is a good approximation if the true $p_T(\mathbf{R}_{\mathrm{FG}} \mid \mathbf{R}_{\mathrm{CG}})$ distribution indeed concentrates at the determined $\mathbf{R}_{\mathrm{FG}}$ value; otherwise (e.g., there are very flexible residue or in relatively high temperature), a more precise coarse-graining method for the energy function is required (e.g. [111]).

Moreover, in PIDP training Eq. (4), although the samples $\{\mathbf{R}_{\mathcal{D},0}^{(m)}\}_{m=1}^{M}$ for evaluating the loss can be taken as any that are relevant to the problem in principle, overly loosely chosen structures may cause numerical difficulties as the corresponding gradient energy gradient would be too large. This is the limiting issue from using normal mode perturbed structures hence we have to resort to MD structures. A possible approach to mitigate this limitation is using a "milder" energy function, which does not increase its value as steeply on off-equilibrium structures. For PIDP training, the energy function only needs to indicate a very small probability, and it does not matter much how small it is, as all small values almost equally indicate a vacuum.

F.3 Limitations on Model Architecture and Scale

Another crucial limitation of DiG is the model restriction, which is resulted from the compromise between the model capacity and the required computational resource. The model capacity determines the expressiveness and generalization ability of the deep learning models for equilibrium distribution prediction, while the computational resource determines the availability and speed of the training and inference processes. In this work, we have to face the constraint of the computational resource and make some choices that may affect the performance of DiG.

For example, for the protein systems, we use a 12-layer Graphormer with about 80M learnable parameters, which is relatively small considering the complexity and diversity of protein structures and distributions. The model capacity of DiG may not be enough to capture the intricate and high-dimensional energy landscapes and distributions of protein systems, and this can be evidenced by the structural quality of the generated protein structures. We observe that smaller models are easily outperformed by larger models. For example, a 4-layer Graphormer with about 10M learnable parameters only produces a median TM-socre [91] of 0.46 on the PDB validation dataset, while a 12-layer Graphormer can easily reach more than 0.8.

Besides, we fix the parameters of the pre-trained Evoformer module in AlphaFold, which is used to extract the features from the protein sequence and the MSA. Evoformer is a powerful and sophisticated module that can encode rich and informative features for protein structure prediction, but it is also computationally expensive and complex, and hasn't been fine-tuned during the training of DiG for predicting equilibrium distribution. This may lead to a significant performance drop since the frozen parameters of Evoformer restrict the expressiveness of DiG very much. In Supplementary Fig. S1a and S1b, the high-density regions of MD simulation are perfectly aligned with both known structures and predicted structures by AlphaFold, but there is an observable shift of the high-density regions generated by DiG. We suspect that the shift is due to the limitation of model capacity. Specifically, a fixed Evoformer implemented in DiG does not perform as well as a learnable Evoformer that is used in AlphaFold. A similar observation is that, although for RBD protein, both AlphaFold and DiG can generate high-quality structures with TMscores > 0.8

in all cases, their performances are different for the case of main protease, where AlphaFold could generate high-quality structures, but structures with TMscores > 0.8 generated by DiG only accounted for about 6.8% of all generated structures (The average TMscore of all generated structures of main protease by DiG is about 0.64). This performance gap between AlphaFold and DiG can be possibly caused by the fixed Evoformer. If so, the performance of DiG could be significantly improved if we can fine-tune the Evoformer module with the data and objective of DiG.

Moreover, in this work, we mainly focus on algorithm development, but not on model architecture development. We mainly use the existing deep learning architectures, such as Graphormer and Evoformer. More advanced and specialized architectures that can better exploit the 3D conformational information and the physics principles of molecular systems may improve the performance and efficiency of DiG.

Therefore, we acknowledge that the model architecture restriction is a serious limitation for DiG in the current implementation, and this will be resolved in the future with enhanced capacity of advanced models.